(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
6 September 2002 (06.09.2002)

PCT

(10) International Publication Number
WO 02/069121 A1

(51) International Patent Classification⁷: G06F 3/00, H04N 7/00

(21) International Application Number: PCT/IL02/00144

(22) International Filing Date: 25 February 2002 (25.02.2002)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
60/272,512 26 February 2001 (26.02.2001) US

(71) Applicant *(for all designated States except US)*: IP PLANET NETWORKS LTD. [IL/IL]; 7 Rival Street, 67778 Tel Aviv (IL).

(72) Inventors; and

(75) Inventors/Applicants *(for US only)*: GISSIN, Yaron [IL/IL]; 6 Sanhedrin Street, Apt. 8, 62916 Tel Aviv (IL). GISSIN, Erez [IL/IL]; 19 Modi'in Street, 47290 Ramat Hasharon (IL). ELGAVISH, Yuval [IL/IL]; 14 Hamaayan Street, 53375 Givataim (IL). NOAM, Johnny [IL/IL]; c/o IP Planet Networks Ltd., 7 Rival Street, 6th Floor, 67778 Tel Aviv (IL).

(74) Agents: SANFORD T. COLB & CO. et al.; P.O. Box 2273, 76122 Rehovot (IL).

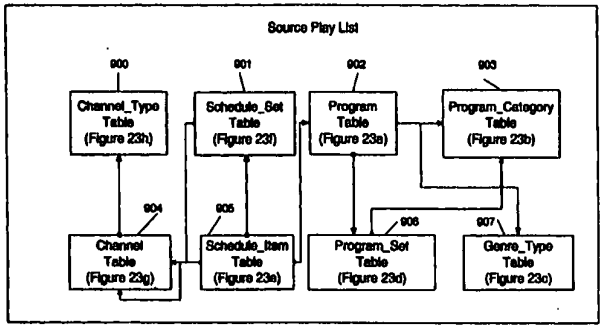
(81) Designated States *(national)*: AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, BZ, CA, CH, CN, CO, CR, CU, CZ, DE, DK, DM, DZ, EC, EE, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MA, MD, MG, MK, MN, MW, MX, MZ, NO, NZ, OM, PH, PL, PT, RO, RU, SD, SE, SG, SI, SK, SL, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VN, YU, ZA, ZM, ZW.

(84) Designated States *(regional)*: ARIPO patent (GH, GM, KE, LS, MW, MZ, SD, SL, SZ, TZ, UG, ZM, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE, TR), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG).

Published:
— *with international search report*
— *before the expiration of the time limit for amending the claims and to be republished in the event of receipt of amendments*

For two-letter codes and other abbreviations, refer to the "Guidance Notes on Codes and Abbreviations" appearing at the beginning of each regular issue of the PCT Gazette.

(54) Title: MODULAR INTERACTIVE APPLICATION GENERATION SYSTEM



(57) Abstract: A system for generating interactive television programs including an interactive item scheduler operative to generate an interactive item schedule (905) for incorporation into at least one television program (902), the interactive item schedule comprising a first plurality of interactive items each associated with a time-stamp, and an interactive television program integrator operative to incorporate the first plurality of interactive items into at least one television program in accordance with the schedule (906).

WO 02/069121 A1

MODULAR INTERACTIVE APPLICATION GENERATION SYSTEM

FIELD OF THE INVENTION

The present invention relates to apparatus and methods for presenting interactive applications.

BACKGROUND OF THE INVENTION

Conventional interactive application generation is described in the following US Patents: 5,861,881; 6,215,484; 6,018,768; 5,861,881; 5,778,181; 5,774,664; 5,632,007; 5,585,858; 4,847,699; 5,537,141; 4,847,698; 5,682,196; 4,847,700; 4,918,516; RE34,340; 5,929,850; 5,682,196; 5,724,091; and 5,848,352.

The disclosures of all publications mentioned in the specification and of the publications cited therein are hereby incorporated by reference.

SUMMARY OF THE INVENTION

The present invention seeks to provide a modular interactive application generation system.

There is thus provided in accordance with a preferred embodiment of the present invention a system for generating interactive television programs including an interactive item scheduler operative to generate an interactive item schedule for incorporation into at least one television program, the interactive item schedule including a first plurality of interactive items each associated with a time-stamp, and an interactive television program integrator operative to incorporate the first plurality of interactive items into at least one television program in accordance with the schedule.

Further in accordance with a preferred embodiment of the present invention, the interactive television program integrator is operative to receive, for each individual one of at least one television programs, an on-air signal indicating, in real-time, the time at which broadcast of the individual television program began.

Still further in accordance with a preferred embodiment of the present invention, the interactive television program integrator is also operative to receive, in advance of broadcast, from an external source, a playlist including a second plurality of

Best Available Copy

television programs to be broadcast and to generate, off-line, an output instruction to a broadcasting facility describing how to incorporate the first plurality of interactive items into the second plurality of television programs in accordance with the schedule.

Additionally in accordance with a preferred embodiment of the present invention, the system also includes an interactive television GUI operative to generate a graphic display of the playlist and of a library of interactive items and to accept an editor-user's input associating an individual interactive item from the library with a temporal location on the playlist.

Still further in accordance with a preferred embodiment of the present invention, the graphic display also includes a video window which, responsive to a user's indication of a temporal location on the playlist, presents a portion of a program associated with the temporal location.

Additionally in accordance with a preferred embodiment of the present invention, the video window, responsive to an editor-user's input associating an individual interactive item from the library with a temporal location on the playlist, presents a portion of a program associated with the temporal location and, concurrently, the portion of the individual interactive item associated with the temporal location.

Further in accordance with a preferred embodiment of the present invention, the interactive television program integrator is operative to display the first plurality of interactive items concurrently with a corresponding first plurality of portions of at least one television program in accordance with the schedule.

Still further in accordance with a preferred embodiment of the present invention, the interactive television program integrator is operative to superimpose at least one of the first plurality of interactive items onto at least one of the corresponding first plurality of portions of at least one television program in accordance with the schedule.

Further in accordance with a preferred embodiment of the present invention, the interactive item scheduler includes an interactive item generator operative to generate at least one interactive item for inclusion in the interactive item schedule.

Additionally in accordance with a preferred embodiment of the present invention, the interactive item generator includes a library of empty interactive item templates and a template filling user interface operative to accept, from an editor-user,

interactive content to fill an editor-user-selected one of the interactive item templates.

Further in accordance with a preferred embodiment of the present invention, the system includes a repository for filled interactive item templates thereby to enable an editor-user to fill templates off-line for real time incorporation into at least one television program.

Still further in accordance with a preferred embodiment of the present invention, at least one time-stamp for at least one individual interactive item includes an absolute time for broadcast of the individual interactive item.

Additionally in accordance with a preferred embodiment of the present invention, at least one time-stamp for at least one individual interactive item includes a time for broadcast of the individual interactive item, relative to an on-air signal to be received which will indicate the time at which broadcast of an individual television program began.

Also provided, in accordance with another preferred embodiment of the present invention, is a methodology for providing enhanced television type content to a plurality of disparate displays including providing television type content, enhancing the television type content in a display-independent manner to provide enhanced display-independent interactive television type content, and providing a plurality of display specific additions to said enhanced display-independent television type content.

Additionally in accordance with a preferred embodiment of the present invention, the methodology includes broadcasting the enhanced display-independent television type content with at least one display specific addition.

Additionally in accordance with a preferred embodiment of the present invention, the methodology also includes receiving and displaying, at a given one of the plurality of disparate displays, the enhanced display-independent television type content with at least one display specific addition.

Also provided, in accordance with a preferred embodiment of the present invention, is a system for authoring and broadcasting of interactive content, the system including creation of interactive content by non-programmers including at least one of the following editing functions: drag-and-drop function for incorporation of interactive content into a program schedule, wizard-based content creation for interactive content, and editing-level synchronization with broadcasting events including a synchronization

information display for the non-programmer interactive content creator.

Additionally provided, in accordance with another preferred embodiment of the present invention, is an interactive content screen display apparatus including a first video area portion displaying a video broadcast, a second interactive portion displaying interactive content selected by a viewer, and a third pushed interrupt portion, which cannot be overridden by the viewer, displaying interrupting interactive content pushed by an interactive content provider, and wherein the second interactive portion cannot be overridden by the interactive content provider.

There is also provided, in accordance with another preferred embodiment of the present invention, a system for conveying interactive content to a plurality of user terminals having different characteristics, the system including a interactive content generator and a plurality of user-terminal specific compilers operative to compile interactive content generated by the interactive content generator so as to adapt the interactive content for use by a corresponding one of the user terminals, thereby to provide interactive content generated by the interactive content generator to all of the plurality of user terminals despite their different characteristics.

Further in accordance with a preferred embodiment of the present invention, the user terminals differ with respect to at least one of the following types of terminal characteristics: user terminal operating system characteristics, user terminal output characteristics, and user terminal input characteristics.

Still further in accordance with a preferred embodiment of the present invention, the interactive content generator includes a library of templates, each template being operative to prompt a content editor to fill the template with specific content, thereby to generate a template instance including an action.

Additionally in accordance with a preferred embodiment of the present invention, each template is operative to prompt the content editor to define a template instance trigger thereby to generate an assigned action.

There is also provided, in accordance with another preferred embodiment of the present invention, an interactive content generation system including an interactive content template repository storing a plurality of templates for interactive content items, and a template filling interface allowing a user to select, view and fill in a template from among the plurality of templates.

It is appreciated that the term "program" is used herein to refer both to entertainment programs (programs which viewers are inherently motivated to watch such as newsshows, sitcoms, quizzes, talkshows, ceremonies and sportscasts) and to commercial programs ("commercials" or programs paid for by external advertisers).

The term "television" or "televised programming" is used herein to refer to any broadcasted content intended for consumption by a remote population of users having receivers such as broadcasted video content for consumers having display screens (also termed herein viewsers) or such as broadcasted audio content for users having audio receivers.

The following terms are used to include the following meanings and can be but are not necessarily limited thereto:

Absolute Time: A time not related to any parameter.

Absolute triggers (time): The time of the trigger is related to the time line to a specific time regardless to any program ID.

Application Composer: A development environment that allows programmers to develop Interactive Application Templates that can then be embedded in the System BackEngine. The environment is XML based, uses IADL (see Figs. 72-95) and features the Application Builder (See Figs. 96 and 97) - a graphic tool for application construction.

Action: An interactive Application Template that an editor-user has filled with data and has not yet assigned a schedule. It is registered in the BackEngine as an action and is ready to be assigned to a time. A simplified graphic representation of an action is described in Figs. 46-50.

Application Builder: A graphic software tool within the Application Composer environment, that utilizes IADL tags to allow computer programmers to construct Interactive Application Templates. It also provides the tools to automatically generate a population wizard. A simplified graphic representation can be found in Figs. 96-97.

Application Loader: A system component that is generated in the DTV Packaging Subsystem (Fig. 13), and is responsible to load an interactive application. Can be referenced as "Notifier" or a graphic representation of a "call for action".

BackEngine: A set of system components that serve as the foundation

elements and the building blocks of the system. The BackEngine is best described in Fig. 4 and also features communication servers, internal management tools, and business logic.

Creative Brief: A document containing concept description, content specifications and requirements, as the initial step of the conceptualization of any interactive application towards implementation.

Current running Schedule Item ID: is embedded in the On-Air signal. The ID relates to a segment of a complete program (probably broken in time by commercials or other programming).

Data Carousel: A server-based software mechanism that broadcasts data to multiple recipients (usually Digital TV Set Top Box subscribers), without the need to employ an open communication line back from the client to the server

Designer: A graphic Designer that is responsible for the Look& Feel of an Interactive Application that runs on a given TV or PC screen.

DTV (Digital Television) Packaging Subsystem: A system component that utilizes the (evolving) Knowledgebase in order to convert and package a system application to a target platform code application before it is sent to the end users.

Editor-User: The person that uses the System Editor-GUI to manage and edit the content and make synchronization decisions on the playlist.

Fuser: A System component that is responsible for blending and integrating an Interactive Application (Created by Target Platform Code) or Interactive Template Application (Created by Application Composer in IADL) - to the System's BackEngine.

IADL: Interactive Application Definition Language: An XML based markup language that allows the construction of basic Interactive Applications, and tools to manage these applications.

Instance: (Also referred to as "Assigned Action")An Interactive Application Template that an editor-user has filled with data and assigned a schedule. It is registered in the BackEngine as an instance and is waiting for its time signal in order to be sent.

Interactive Application: An application that allows end users (Viewsers) to interact, via PC or Digital TV with pre-defined activities.

Interactive Application Template: An application template that resides in the BackEngine database and is ready to be used by the editor-user with the Editor-GUI.

Interactive Application Template Files: XML/HTML or Platform target code and Media and Resource Files.

Interactive Message: In IP environment, the "call for action" - text based data that is embedded with the triggers in the video, and is displayed at the correct time to the eyes of the end user (Viewser).

Interactive scheduled Application: An instance that has been compiled, converted to the target platform code and sent to the Broadcasting Gateway. Interactive Scheduled Application Usually equals Application Loader + Interactive Application.

Media Highway: A middleware platform for interactive broadcasting produced by Canal + Technologies.

Middleware: Software that runs on a Set Top Box and is responsible for the communications of interactive applications, the Box and the Remote control. Also hosts the software and resources to run the Conditional Access system and the return channel in the set top box.

Notifier: In a DTV environment, the " call for action" - data that is embedded with the triggers in the video, and is displayed at the correct time to the eyes of the end user (Viewser). (e.g. Sky Active Red Button display for interactive).

On-Air signal: Indicates the actual start time of a program in real time.

On Air message: System formatted On-Air signal containing Time of arrival - Relative time; and the Current running program's ID.

OpenTV: A company and its related middleware technology platform that enables Interactive broadcasting to digital TV customers.

Playlist: A concurrent list of programs and programs segments usually generated by broadcasting scheduling systems

Population Wizard: An interface that allows editor-users to incorporate new data, or update existing data in an interactive application template.

Programmer: A software developer.

Relative triggers (time): The time of the trigger is related to the start time of program ID in the time line.

Return channel (back path): A general term for communications from the Digital TV (SetTop Box) subscriber to the broadcasting operators systems (headend).

SetTop Box: A computing device installed at digital TV subscribers that enables the reception and presentation of digital video, audio and interactive applications on a TV screen. Usually interfaces with the household TV Remote Control.

Editor-GUI: A set of programs that present a usable man-machine interfaces to allow editors-users to manage, store and edit the data that runs through the system.

System Parameters: Pre-defined (could be changed) system parameters for various components in the system. Simple text file containing the name of the field and its value.

Target Platform Code: A development language related to a specific third-party broadcasting technology. (e.g. OpenTV, MediaHighway).

Thin Client: A wrapper application that is attached to an Interactive Scheduled Application that is sent to the Set Top Box.

Trigger ID: A unique number that relates to a specific action and all its related data.

TIM - Trigger Insertion Mechanism: A system Component that is responsible for the timed delivery of Scheduled interactive applications.

Trigger sliding window duration: the TIM Server collects future triggers according to this "window" of time.

Viewser: Viewer + User. End users using a TV or a PC device that are the recipients of the content generated by the system. A Viewser is a person that interacts with a given TV program.

Video Feed: A televised broadcasting signal containing video content.

BRIEF DESCRIPTION OF THE DRAWINGS

The present invention will be understood and appreciated from the following detailed description, taken in conjunction with the drawings in which:

Fig. 1 is a simplified functional block diagram of a system for incorporating interactive applications into video programming, the interactive applications being adapted for display and interaction within a computer networked to

the system, the system being constructed and operative in accordance with a first preferred embodiment of the present invention;

Fig. 2 is a simplified functional block diagram of a system for incorporating interactive applications into video programming, the interactive applications being adapted for display and interaction within a digital television set associated via a digital television network with the system, the system being constructed and operative in accordance with a first preferred embodiment of the present invention;

Fig. 3 is a simplified functional block diagram of a system for incorporating interactive applications into video programming, the interactive applications being adapted for display and interaction within one or more of the following: a computer networked to the system, and/or a digital television set associated via a digital television network with the system, the system being constructed and operative in accordance with a first preferred embodiment of the present invention;

Fig. 4 is a simplified functional block diagram of the BackEngine of Figs. 1 - 3, constructed and operative in accordance with a preferred embodiment of the present invention;

Fig. 5 is a simplified functional block diagram of the application protocol interfaces block of Figs. 1 - 3, constructed and operative in accordance with a preferred embodiment of the present invention;

Fig. 6 is a simplified functional block diagram of the application composer of Figs. 1 - 3, constructed and operative in accordance with a preferred embodiment of the present invention;

Fig. 7 is a simplified functional block diagram of the IP broadcast gateway of Figs. 1 and 3, constructed and operative in accordance with a preferred embodiment of the present invention;

Fig. 8 is a simplified functional block diagram of the DTV broadcast gateway of Figs. 2 and 3, constructed and operative in accordance with a preferred embodiment of the present invention;

Fig. 9 is a simplified functional block diagram of the application fuser of Figs. 1 - 3, constructed and operative in accordance with a preferred embodiment of the present invention;

Fig. 10 is a simplified functional block diagram of the interactive server

of Figs. 1 - 3, constructed and operative in accordance with a preferred embodiment of the present invention;

Fig. 11 is a simplified functional block diagram of the thin client of Figs. 2 - 3, constructed and operative in accordance with a preferred embodiment of the present invention;

Fig. 12 is a simplified functional block diagram of the feedback system of Figs. 1 - 3, constructed and operative in accordance with a preferred embodiment of the present invention;

Fig. 13 is a simplified functional block diagram of the DTV packager of Fig. 4, constructed and operative in accordance with a preferred embodiment of the present invention;

Fig. 14 is a simplified functional block diagram of the BackEngine database of Fig. 4, constructed and operative in accordance with a preferred embodiment of the present invention;

Fig. 15 is a simplified illustration of relationships between the tables of the source playlist of Fig. 14, constructed and operative in accordance with a preferred embodiment of the present invention;

Fig. 16 is a simplified illustration of relationships between the tables of the program categories block of Fig. 14, constructed and operative in accordance with a preferred embodiment of the present invention;

Fig. 17 is a simplified illustration of relationships between the tables of the personalization block of Fig. 14, constructed and operative in accordance with a preferred embodiment of the present invention;

Fig. 18 is a simplified illustration of relationships between the tables of the activities logs block of Fig. 14, constructed and operative in accordance with a preferred embodiment of the present invention;

Fig. 19 is a simplified illustration of relationships between the tables of the monitoring and control block of Fig. 14, constructed and operative in accordance with a preferred embodiment of the present invention;

Fig. 20 is a simplified illustration of relationships between the tables of the users table cluster of Fig. 14, constructed and operative in accordance with a preferred embodiment of the present invention;

Fig. 21 is a simplified illustration of relationships between the tables of the interactive message repository of Fig. 14, constructed and operative in accordance with a preferred embodiment of the present invention;

Fig. 22 is a simplified illustration of relationships between the tables of the application repository of Fig. 14, constructed and operative in accordance with a preferred embodiment of the present invention;

Fig. 23A is a diagram illustrating the data included in the program table of Fig. 15, in accordance with a preferred embodiment of the present invention;

Fig. 23B is a diagram illustrating the data included in the program category table of Fig. 15, in accordance with a preferred embodiment of the present invention;

Fig. 23C is a diagram illustrating the data included in the genre type table of Fig. 15, in accordance with a preferred embodiment of the present invention;

Fig. 23D is a diagram illustrating the data included in the program set table of Fig. 15, in accordance with a preferred embodiment of the present invention;

Fig. 23E is a diagram illustrating the data included in the schedule item table of Fig. 15, in accordance with a preferred embodiment of the present invention;

Fig. 23F is a diagram illustrating the data included in the schedule set table of Fig. 15, in accordance with a preferred embodiment of the present invention;

Fig. 23G is a diagram illustrating the data included in the channel table of Fig. 15, in accordance with a preferred embodiment of the present invention;

Fig. 23H is a diagram illustrating the data included in the channel type table of Fig. 15, in accordance with a preferred embodiment of the present invention;

Fig. 24A is a diagram illustrating the data included in the program categories binding table of Fig. 16, in accordance with a preferred embodiment of the present invention;

Fig. 24B is a diagram illustrating the data included in the subcategory definition (sub_category_def) table of Fig. 16, in accordance with a preferred embodiment of the present invention;

Fig. 24C is a diagram illustrating the data included in the program category definition table of Fig. 16, in accordance with a preferred embodiment of the present invention;

Fig. 25A is a diagram illustrating the data included in the viewsers table of Fig. 17, in accordance with a preferred embodiment of the present invention;

Fig. 25B is a diagram illustrating the data included in the occupation table of Fig. 17, in accordance with a preferred embodiment of the present invention;

Fig. 25C is a diagram illustrating the data included in the region table of Fig. 17, in accordance with a preferred embodiment of the present invention;

Fig. 25D is a diagram illustrating the data included in the age table of Fig. 17, in accordance with a preferred embodiment of the present invention;

Fig. 25E is a diagram illustrating the data included in the interest table of Fig. 17, in accordance with a preferred embodiment of the present invention;

Fig. 25F is a diagram illustrating the data included in the industry table of Fig. 17, in accordance with a preferred embodiment of the present invention;

Fig. 25G is a diagram illustrating the data included in the country table of Fig. 17, in accordance with a preferred embodiment of the present invention;

Fig. 25H is a diagram illustrating the data included in the comments-on-viewser table of Fig. 17, in accordance with a preferred embodiment of the present invention;

Fig. 25I is a diagram illustrating the data included in the pilot control center emails table of Fig. 17, in accordance with a preferred embodiment of the present invention;

Fig. 25J is a diagram illustrating the data included in the connection table of Fig. 17, in accordance with a preferred embodiment of the present invention;

Fig. 26A is a diagram illustrating the data included in the schedule items logs table of Fig. 18, in accordance with a preferred embodiment of the present invention;

Fig. 26B is a diagram illustrating the data included in the program log table of Fig. 18, in accordance with a preferred embodiment of the present invention;

Fig. 26C is a diagram illustrating the data included in the viewsers activities logs table of Fig. 18, in accordance with a preferred embodiment of the present invention;

Fig. 26D is a diagram illustrating the data included in the application result table of Fig. 18, in accordance with a preferred embodiment of the present

invention;

Fig. 26E is a diagram illustrating the data included in the error table of Fig. 18, in accordance with a preferred embodiment of the present invention;

Fig. 27A is a diagram illustrating the data included in the system parameters table of Fig. 14, in accordance with a preferred embodiment of the present invention;

Fig. 27B is a diagram illustrating the data included in the module parameters table of Fig. 14, in accordance with a preferred embodiment of the present invention;

Fig. 28A is a diagram illustrating the data included in the trigger table of Fig. 14, in accordance with a preferred embodiment of the present invention;

Fig. 28B is a diagram illustrating the data included in the trigger template table of Fig. 14, in accordance with a preferred embodiment of the present invention;

Fig. 29A is a diagram illustrating the data included in the control center users table of Fig. 20, in accordance with a preferred embodiment of the present invention;

Fig. 29B is a diagram illustrating the data included in the system users table of Fig. 20, in accordance with a preferred embodiment of the present invention;

Fig. 29C is a diagram illustrating the data included in the permission level table of Fig. 20, in accordance with a preferred embodiment of the present invention;

Fig. 29D is a diagram illustrating the data included in the user password history table of Fig. 20, in accordance with a preferred embodiment of the present invention;

Fig. 30A is a diagram illustrating the data included in the interactive message template data table of Fig. 21, in accordance with a preferred embodiment of the present invention;

Fig. 30B is a diagram illustrating the data included in the interactive message type table of Fig. 21, in accordance with a preferred embodiment of the present invention;

Fig. 30C is a diagram illustrating the data included in the interactive message template table of Fig. 21, in accordance with a preferred embodiment of the

present invention;

Fig. 30D is a diagram illustrating the data included in the interactive message instance data table of Fig. 21, in accordance with a preferred embodiment of the present invention;

Fig. 30E is a diagram illustrating the data included in the interactive message instance data table of Fig. 21, in accordance with a preferred embodiment of the present invention;

Fig. 31A is a diagram illustrating the data included in the application template table of Fig. 22, in accordance with a preferred embodiment of the present invention;

Fig. 31B is a diagram illustrating the data included in the application template data table of Fig. 22, in accordance with a preferred embodiment of the present invention;

Fig. 31C is a diagram illustrating the data included in the application type table of Fig. 22, in accordance with a preferred embodiment of the present invention;

Fig. 31D is a diagram illustrating the data included in the application instance table of Fig. 22, in accordance with a preferred embodiment of the present invention;

Fig. 31E is a diagram illustrating the data included in the application instance data table of Fig. 22, in accordance with a preferred embodiment of the present invention;

Fig. 31F is a diagram illustrating the data included in the graphic skin table of Fig. 22, in accordance with a preferred embodiment of the present invention;

Fig. 32 is a diagram illustrating the data included in the eplaylist table of Fig. 14, in accordance with a preferred embodiment of the present invention;

Fig. 33 is a simplified illustration of relationships between the tables of the knowledgebase of Fig. 4, constructed and operative in accordance with a preferred embodiment of the present invention;

Fig. 34A – 34B, taken together, form a table summarizing the input parameters, output parameters and preferred mode of operation for wizard processing procedures useful in manipulating wizard data within the BackEngine database of Figs.

4 and 14;

Figs. 35A - 35B, taken together, form a table summarizing the input parameters, output parameters and preferred mode of operation for trigger processing procedures useful in manipulating trigger data within the BackEngine database of Figs. 4 and 14;

Fig. 36 is a table summarizing the input parameters, output parameters and preferred mode of operation for repository maintenance procedures useful in manipulating repository data within the BackEngine database of Figs. 4 and 14;

Fig. 37 is a table summarizing the input parameters, output parameters and preferred mode of operation for playlist processing procedures useful in manipulating playlist data within the BackEngine database of Figs. 4 and 14;

Fig. 38 is a table summarizing the input parameters, output parameters and preferred mode of operation for digital television trigger insertion procedures useful in manipulating data within the BackEngine database of Figs. 4 and 14 in accordance with the embodiments of Figs. 2 and 3;

Fig. 39 is a table summarizing the input parameters, output parameters and preferred mode of operation for packager procedures performed by the DTV packager of Figs. 4 and 12;

Fig. 40 is a table summarizing the input parameters, output parameters and preferred mode of operation for viewser log procedures useful in manipulating data within the backengine database of Figs. 4 and 14;

Fig. 41 is a table summarizing the input parameters, output parameters and preferred mode of operation for fusing procedures performed by the fuser of Figs. 4 and 9;

Fig. 42 is a table summarizing the input parameters, output parameters and preferred mode of operation for interactive serving procedures performed by the interactive server of Figs. 4 and 10;

Fig. 43 is a table summarizing the input parameters, output parameters and preferred mode of operation for computer network trigger insertion procedures useful in manipulating data within the BackEngine database of Figs. 4 and 14 in accordance with the embodiments of Figs. 1 and 3;

Fig. 44 is a table summarizing the input parameters, output parameters

and preferred mode of operation for procedures useful in manipulating application instance data within the application repository of Fig. 22;

Fig. 45A – 45B, taken together, form a table summarizing the input parameters, output parameters and preferred mode of operation for application protocol interfacing procedures useful in manipulating data within the backengine database of Figs. 4 and 14;

Fig. 46 is a simplified pictorial illustration of a first screen display generated by the editor GUI (graphic user interface) of Fig. 4 when performing an example workflow;

Fig. 47 is a simplified pictorial illustration of a second screen display generated by the editor GUI (graphic user interface) of Fig. 4 when performing an example workflow;

Fig. 48 is a simplified pictorial illustration of a third screen display generated by the editor GUI (graphic user interface) of Fig. 4 when performing an example workflow;

Fig. 49 is a simplified pictorial illustration of a fourth screen display generated by the editor GUI (graphic user interface) of Fig. 4 when performing an example workflow;

Fig. 50 is a simplified pictorial illustration of a fifth screen display generated by the editor GUI (graphic user interface) of Fig. 4 when performing an example workflow;

Fig. 51 is a simplified flowchart illustration of a preferred method of operation for the system of Fig. 1.

Fig. 52 is a simplified flowchart illustration of a preferred method of operation for the system of Fig. 2.

Figs. 53A - 53B, taken together, form a simplified flowchart illustration of a preferred method of operation for the BackEngine of Figs. 1, 2 and 4;

Fig. 54 is a diagram showing a typical life-cycle of a modular interactive application in accordance with a preferred embodiment of the present invention;

Fig. 55 is a simplified flowchart illustration of a preferred method by which application composer 170 of Figs. 1 and 6 performs the interactive application file generation step 1010 of Fig. 51;

Fig. 56 is a simplified flowchart illustration of a preferred method by which fuser 110 of Figs. 1 and 9 performs the interactive application template generation step 1030 of Fig. 51;

Figs. 57A - 57B, taken together, form an example of a main.XML file that describes the syntax of any interactive application file generated by the application composer 170;

Fig. 58 is a simplified flowchart illustration of a preferred method whereby the editor GUI 280 in BackEngine 100 of Figs. 1 and 4 perform a first part of the interactive scheduled application generation step 1040 of Fig. 51, corresponding to content injection step 1190 in Fig. 53A;

Fig. 59 is a simplified flowchart illustration of a preferred method whereby the editor GUI 280 in BackEngine 100 of Figs. 1 and 4 perform a second part of the interactive scheduled application generation step 1040 of Fig. 51, corresponding to assignment to timeline step 1210 in Fig. 53A;

Fig. 60 is a simplified flowchart illustration of a preferred method whereby the BackEngine of Figs. 1 and 4 performs the interactive scheduled application step 1044 of Fig. 51;

Fig. 61 is a simplified flowchart illustration of a preferred method whereby the BackEngine 100 of Figs. 1 and 4 performs the on-air signal receiving step 1046 of Fig. 51;

Fig. 62 is a simplified flowchart illustration of a preferred method whereby the IP broadcast gateway 201 of Figs. 1 and 7 performs the interactive scheduled application broadcasting step 1050 of Fig. 51;

Fig. 63 is a simplified flowchart illustration of a preferred method whereby the viewser uses his PC to generate a viewser response which is subsequently processed by the system of the present invention in steps 1060 and 1070 of Fig. 51;

Fig. 64 is a simplified flowchart illustration of a preferred method whereby the BackEngine 100 of Figs. 1 and 4 performs the viewser response processing step 1070 of Fig. 51;

Fig. 65 is a simplified flowchart illustration of a preferred method whereby the feedback system 160 of Figs. 1 and 12 performs the viewser response statistics reporting step 1080 of Fig. 51;

Fig. 66A is a simplified flowchart illustration of a preferred method whereby the BackEngine of Figs. 2 and 4 performs the interactive scheduled application sending step 1124 of Fig. 52;

Figs. 66B - 66C, taken together, form a simplified flowchart illustration of a preferred method whereby the DTV packaging subsystem 270 of Figs. 4 and 13 performs the DTV packaging step in the method of Fig. 66A, thereby to generate a packaged instance;

Fig. 66D is a simplified flowchart illustration of a preferred method of operation for the IADL transformer 680 of Fig. 13;

Fig. 67 is a simplified flowchart illustration of a preferred method whereby the DTV broadcast gateway 200 of Figs. 2 and 8 performs the interactive scheduled application broadcasting step 1130 of Fig. 51;

Fig. 68 is a simplified flowchart illustration of a preferred method whereby the DTV broadcast gateway 200 of Figs. 2 and 8 performs the viewser response receiving step 1140 of Fig. 52;

Fig. 69 is a simplified flowchart illustration of a preferred method whereby the viewser uses interactive application digital TV interface software typically residing within his set-top box according to a preferred embodiment of the present invention, to generate a viewser response which is subsequently processed by the system of the present invention in steps 1140 and 1150 of Fig. 52;

Fig. 70 is a simplified flowchart illustration of a preferred method whereby the sync driver 220 of Fig. 4 performs the playlist processing step of Fig. 53A whereby the playlist is prepared for display by GUI 280 of Fig. 4;

Figs. 71A - 71B, taken together, form a simplified flowchart illustration of a preferred method whereby the trigger insertion mechanism server 210 of Fig. 4 performs the interactive scheduled application generation step of Fig. 53B;

Fig. 72A is a table describing two IADL application-level commands having a common syntax;

Fig. 72B is a syntax diagram describing the syntax of each of the commands in Fig. 72A;

Fig. 73A is a table describing four IADL stage-level commands having a common syntax;

Fig. 73B is a syntax diagram describing the syntax of each of the commands in Fig. 73A;

qc Figs. 74A, 75A, 76A, 77A, 78A, 78A, 79A, 80A, 81A, 82A, 83A, 84A, 85A and 86A are tables, each row of which describes an element-level IADL command wherein the commands in each such table have a common syntax;

Figs. 74B, 75B, 76B, 77B, 78B, 78B, 79B, 80B, 81B, 82B, 83B, 84B, 85B and 86B describe the syntaxes of the commands of Figs. 74A, 75A, 76A, 77A, 78A, 78A, 79A, 80A, 81A, 82A, 83A, 84A, 85A and 86A respectively; and

Figs. 87A, 88A, 89A, 90A, 91A, 92A, 93A, 94A and 95A are tables, each row of which describes an atom-level IADL command wherein the commands in each such table have a common syntax;

Figs. 87B, 88B, 89B, 90B, 91B, 92B, 93B, 94B and 95B describe the syntaxes of the commands of Figs. 87A, 88A, 89A, 90A, 91A, 92A, 93A, 94A and 95A respectively;

Fig. 96 is a simplified pictorial illustration of a first screen display generated by the Application Builder 320 and its GUI 330 (Fig. 6) when performing an example workflow as described in Fig. 55. The Application Builder is a preferred tool for the creation of Interactive Application templates; and

Figs. 97A - 97F are simplified pictorial illustrations of the stage(skeleton), project browser, elements inspector, saved elements, functions and atoms library windows, respectively, in the screen display of Fig. 96.

DETAILED DESCRIPTION OF PREFERRED EMBODIMENTS

Fig. 1 is a simplified functional block diagram of a system for incorporating interactive applications into video programming, the interactive applications being adapted for display and interaction within a computer networked to the system, the system being constructed and operative in accordance with a first preferred embodiment of the present invention. A particular feature of a preferred embodiment of the present invention is that the content generation interface is simple enough such that content generation may be performed by human operators with little or no programming experience.

Fig. 2 is a simplified functional block diagram of a system for

incorporating interactive applications into video programming, the interactive applications being adapted for display and interaction within a digital television set associated via a digital television network with the system, the system being constructed and operative in accordance with a first preferred embodiment of the present invention.

Fig. 3 is a simplified functional block diagram of a system for incorporating interactive applications into video programming, the interactive applications being adapted for display and interaction within one or more of the following: a computer networked to the system, and/or a digital television set associated via a digital television network with the system, the system being constructed and operative in accordance with a first preferred embodiment of the present invention.

As shown, the system of Fig. 3 is typically operative in conjunction with conventional home viewer equipment including a PC and a television device having a digital TV decoder also termed herein a "set-top box".

An array of broadcast gateways 200 and 201 typically comprises a broadcast gateway for each of a plurality of interactive content display devices such as one or more television set-top-boxes each running a different operating system, and/or one or more computer devices each having its own computer display format.

Fig. 4 is a simplified functional block diagram of the BackEngine of Figs. 1 - 3, constructed and operative in accordance with a preferred embodiment of the present invention. It is appreciated that the interactive editor-GUI 280 is operative to perform an interactive content generation function and an interactive content incorporation function in which interactive content, once generated, is incorporated into an existing video schedule.

It is appreciated that generally, broadcasting queries to viewsers is less resource-consuming than receiving responses to the queries and processing them. Therefore, according to another preferred embodiment of the present invention, quizzes in which a prize is awarded to the earliest-generated correct response may be presented and the set-top boxes of the viewsers may be operative to store responses and the time at which they were generated using an internal set-top-box clock, synchronized across the population of set-top-boxes, and not send them, pending further instructions. Further instructions may comprise one or more of the following which typically are sent in order:

a. A message indicating that any set-top box storing an answer other than X, which is the correct answer, should destroy the answer and not send it in.

b. A message, typically sent only to a subset of the set-top boxes, indicating that set-top boxes which are storing an X-answer should send in their time of response. The system then typically identifies the earliest time of response.

c. A message, sent either to all set-top boxes or only to a subset thereof, indicating that set-top boxes which are storing an X-answer and a time earlier than the earliest time of response identified in (b), should send in their time of response. The system then typically, as in step (b), identifies the earliest time of response.

d. Step (c) is repeated until, responsive to a message sent to all set-top boxes, no set-top box responds, indicating that the earliest time of response identified by the system in the previous iteration, is the earliest time at which the correct answer was generated.

A particular advantage of a preferred embodiment of the present invention in which templates are used which have features identifiable by the viewser, is that the viewser learns to recognize the various templates and orient himself in the content universe. When the user encounters a template he has seen before, it is easier for the user to assimilate the new information since it is being presented in a familiar format.

A TIM (trigger insertion mechanism) Server 210 synchronizes interactive applications, typically at the sub-program level, to the video. For example, an input to the TIM Server 210 may indicate that an item of interactive content should be incorporated 3 minutes into a program which started 2 minutes ago (according to the on-air signal received from the SyncDriver 220). Therefore, in one minute, the TIM server 210 will generate a command to one or more of the broadcast gateways 200 and/or 201 that the item of interactive content should be introduced.

Fig. 5 is a simplified functional block diagram of the application protocol interfaces block of Figs. 1 - 3, constructed and operative in accordance with a preferred embodiment of the present invention.

Fig. 6 is a simplified functional block diagram of the application composer of Figs. 1 - 3, constructed and operative in accordance with a preferred embodiment of the present invention.

According to a preferred embodiment of the present invention, templates have a life-cycle typically including the following three stages of life:

a. blank; Also termed as "Interactive Application Template"

b. filled-in, also termed herein "Action"

c. assigned, also termed herein "Instance"

Examples of Interactive Application Templates include the following:

a. A buying template in which a viewser is asked to purchase a product;

b. A product information template in which product information is displayed to a viewser;

c. A survey template in which viewsers are invited to take part in a survey, typically by manually or orally keying in a response to one or more multiple choice questions;

d. A "did you know" template in which viewsers are invited to be exposed to additional information about a topic of presumed interest;

e. A "breaking news" template which displays breaking news;

f. An "internal html page" template, intended for viewsers using a PC screen rather than a television screen, in which an html page pops up interactively within a video program;

g. An "external html page"

h. A "trivia" template which invites a user to participate in a trivia quiz;

i. A "survey results" template, which displays results of a survey, typically an interactive survey which may have been presented using the above-described "survey" template;

j. A "decision simulation" template, prompting the viewser to put himself in the place of a newsmaker and determine what he would decide if he were in the newsmaker's position;

k. A "now showing" template inviting the viewser to view information about a current program;

l. A "promotion" template inviting the viewser to view information about a future program;

m. A "where you can find me" template inviting the viewser to enter particulars regarding his location and to receive in return the location of an outlet in

which an advertised product is being sold;

n. A "show merchandise" template inviting the user to receive information regarding products pertinent to the program currently on air.

Fig. 7 is a simplified functional block diagram of the IP broadcast gateway of Figs. 1 and 3, constructed and operative in accordance with a preferred embodiment of the present invention.

Fig. 8 is a simplified functional block diagram of the DTV broadcast gateway of Figs. 2 and 3, constructed and operative in accordance with a preferred embodiment of the present invention.

Fig. 9 is a simplified functional block diagram of the application fuser of Figs. 1 - 3, constructed and operative in accordance with a preferred embodiment of the present invention.

Fig. 10 is a simplified functional block diagram of the interactive server of Figs. 1 - 3, constructed and operative in accordance with a preferred embodiment of the present invention.

Fig. 11 is a simplified functional block diagram of the thin client of Figs. 2 - 3, constructed and operative in accordance with a preferred embodiment of the present invention.

Fig. 12 is a simplified functional block diagram of the feedback system of Figs. 1 - 3, constructed and operative in accordance with a preferred embodiment of the present invention.

Fig. 13 is a simplified functional block diagram of the DTV packager of Fig. 4, constructed and operative in accordance with a preferred embodiment of the present invention.

Fig. 14 is a simplified functional block diagram of the BackEngine database of Fig. 4, constructed and operative in accordance with a preferred embodiment of the present invention.

Fig. 15 is a simplified illustration of relationships between the tables of the source playlist of Fig. 14, constructed and operative in accordance with a preferred embodiment of the present invention.

Fig. 16 is a simplified illustration of relationships between the tables of the program categories block of Fig. 14, constructed and operative in accordance with a

preferred embodiment of the present invention.

Fig. 17 is a simplified illustration of relationships between the tables of the personalization block of Fig. 14, constructed and operative in accordance with a preferred embodiment of the present invention.

Fig. 18 is a simplified illustration of relationships between the tables of the activities logs block of Fig. 14, constructed and operative in accordance with a preferred embodiment of the present invention.

Fig. 19 is a simplified illustration of relationships between the tables of the optional monitoring and control block of Fig. 14, constructed and operative in accordance with a preferred embodiment of the present invention.

Fig. 20 is a simplified illustration of relationships between the tables of the users table cluster of Fig. 14, constructed and operative in accordance with a preferred embodiment of the present invention.

Fig. 21 is a simplified illustration of relationships between the tables of the interactive message repository of Fig. 14, constructed and operative in accordance with a preferred embodiment of the present invention.

Fig. 22 is a simplified illustration of relationships between the tables of the application repository of Fig. 14, constructed and operative in accordance with a preferred embodiment of the present invention.

Fig. 23A is a diagram illustrating the data included in the program table of Fig. 15, in accordance with a preferred embodiment of the present invention.

Fig. 23B is a diagram illustrating the data included in the program category table of Fig. 15, in accordance with a preferred embodiment of the present invention.

Fig. 23C is a diagram illustrating the data included in the genre type table of Fig. 15, in accordance with a preferred embodiment of the present invention.

Fig. 23D is a diagram illustrating the data included in the program set table of Fig. 15, in accordance with a preferred embodiment of the present invention.

Fig. 23E is a diagram illustrating the data included in the schedule item table of Fig. 15, in accordance with a preferred embodiment of the present invention.

Fig. 23F is a diagram illustrating the data included in the schedule set table of Fig. 15, in accordance with a preferred embodiment of the present invention.

Fig. 23G is a diagram illustrating the data included in the channel table of Fig. 15, in accordance with a preferred embodiment of the present invention.

Fig. 23H is a diagram illustrating the data included in the channel type table of Fig. 15, in accordance with a preferred embodiment of the present invention.

Fig. 24A is a diagram illustrating the data included in the program categories binding table of Fig. 16, in accordance with a preferred embodiment of the present invention.

Fig. 24B is a diagram illustrating the data included in the subcategory definition (sub_category_def) table of Fig. 16, in accordance with a preferred embodiment of the present invention.

Fig. 24C is a diagram illustrating the data included in the program category definition table of Fig. 16, in accordance with a preferred embodiment of the present invention.

Fig. 25A is a diagram illustrating the data included in the viewsers table of Fig. 17, in accordance with a preferred embodiment of the present invention.

Fig. 25B is a diagram illustrating the data included in the occupation table of Fig. 17, in accordance with a preferred embodiment of the present invention.

Fig. 25C is a diagram illustrating the data included in the region table of Fig. 17, in accordance with a preferred embodiment of the present invention.

Fig. 25D is a diagram illustrating the data included in the age table of Fig. 17, in accordance with a preferred embodiment of the present invention.

Fig. 25E is a diagram illustrating the data included in the interest table of Fig. 17, in accordance with a preferred embodiment of the present invention.

Fig. 25F is a diagram illustrating the data included in the industry table of Fig. 17, in accordance with a preferred embodiment of the present invention.

Fig. 25G is a diagram illustrating the data included in the country table of Fig. 17, in accordance with a preferred embodiment of the present invention.

Fig. 25H is a diagram illustrating the data included in the comments-on-viewser table of Fig. 17, in accordance with a preferred embodiment of the present invention.

Fig. 25I is a diagram illustrating the data included in the pilot control center emails table of Fig. 17, in accordance with a preferred embodiment of the present

invention.

Fig. 25J is a diagram illustrating the data included in the connection table of Fig. 17, in accordance with a preferred embodiment of the present invention.

Fig. 26A is a diagram illustrating the data included in the schedule items logs table of Fig. 18, in accordance with a preferred embodiment of the present invention.

Fig. 26B is a diagram illustrating the data included in the program log table of Fig. 18, in accordance with a preferred embodiment of the present invention.

Fig. 26C is a diagram illustrating the data included in the viewsers activities logs table of Fig. 18, in accordance with a preferred embodiment of the present invention.

Fig. 26D is a diagram illustrating the data included in the application result table of Fig. 18, in accordance with a preferred embodiment of the present invention.

Fig. 26E is a diagram illustrating the data included in the error table of Fig. 18, in accordance with a preferred embodiment of the present invention.

Fig. 27A is a diagram illustrating the data included in the system parameters table of Fig. 14, in accordance with a preferred embodiment of the present invention.

Fig. 27B is a diagram illustrating the data included in the module parameters table of Fig. 14, in accordance with a preferred embodiment of the present invention.

Fig. 28A is a diagram illustrating the data included in the trigger table of Fig. 14, in accordance with a preferred embodiment of the present invention.

Fig. 28B is a diagram illustrating the data included in the trigger template table of Fig. 14, in accordance with a preferred embodiment of the present invention.

Fig. 29A is a diagram illustrating the data included in the control center users table of Fig. 20, in accordance with a preferred embodiment of the present invention.

Fig. 29B is a diagram illustrating the data included in the system users table of Fig. 20, in accordance with a preferred embodiment of the present invention.

Fig. 29C is a diagram illustrating the data included in the permission level table of Fig. 20, in accordance with a preferred embodiment of the present invention.

Fig. 29D is a diagram illustrating the data included in the user password history table of Fig. 20, in accordance with a preferred embodiment of the present invention.

Fig. 30A is a diagram illustrating the data included in the interactive message template data table of Fig. 21, in accordance with a preferred embodiment of the present invention.

Fig. 30B is a diagram illustrating the data included in the interactive message type table of Fig. 21, in accordance with a preferred embodiment of the present invention.

Fig. 30C is a diagram illustrating the data included in the interactive message template table of Fig. 21, in accordance with a preferred embodiment of the present invention.

Fig. 30D is a diagram illustrating the data included in the interactive message instance data table of Fig. 21, in accordance with a preferred embodiment of the present invention.

Fig. 30E is a diagram illustrating the data included in the interactive message instance data table of Fig. 21, in accordance with a preferred embodiment of the present invention.

Fig. 31A is a diagram illustrating the data included in the application template table of Fig. 22, in accordance with a preferred embodiment of the present invention.

Fig. 31B is a diagram illustrating the data included in the application template data table of Fig. 22, in accordance with a preferred embodiment of the present invention.

Fig. 31C is a diagram illustrating the data included in the application type table of Fig. 22, in accordance with a preferred embodiment of the present invention.

Fig. 31D is a diagram illustrating the data included in the application instance table of Fig. 22, in accordance with a preferred embodiment of the present

invention.

Fig. 31E is a diagram illustrating the data included in the application instance data table of Fig. 22, in accordance with a preferred embodiment of the present invention.

Fig. 31F is a diagram illustrating the data included in the graphic skin table of Fig. 22, in accordance with a preferred embodiment of the present invention.

Fig. 32 is a diagram illustrating the data included in the eplaylist table of Fig. 14, in accordance with a preferred embodiment of the present invention.

Fig. 33 is a simplified illustration of relationships between the tables of the knowledgebase of Fig. 4, constructed and operative in accordance with a preferred embodiment of the present invention.

The tables of the knowledge base illustrated in Fig. 33 may, for example, store the following parameters:

table TextAlign

TextAlign
TextAlignPos

table TextVerticalAlign

TextVerticalAlign
TextVerticalAlignPos

table FontWeight

FontWeight
FontWeightName

table FontStyle

FontStyle
FontStyleName

table TextDecoration

TextDecoration
TextDecorationName

table Display

Display
DisplayName

table ObjectType

Type
TypeName

table Scroll

Scroll
ScrollName

table StageCode

XSLFileName
XSLCode

table ApplicationCode

XSLFileName
XSLCode

table FontFamily

FontFamily
FontFamilyName

table IADLAtoms

AtomID
AtomXML
PCWeb
OpenTV

table IADLElements

ElementID
ElementXML
PCWeb
OpenTV

table IADLAttributs

AttributeID
AttributeName
IADLSyntax
PCWeb
OpenTV

table IADLFunctions

FuncID
FuncSyntax

table ArgumentsType

ArgumentType
Type

table 16FixedColor

CIndex
CName
CPCWeb
COpenTV

table Platform

PlatformID

PlatformName

table AtomList

AtomID
AtomName
PCWeb
OpenTV
Type

table ElementList

ElementID
ElementName
PCWeb
OpenTV
Type

table FunctionList

FuncID
Type
FuncName
PCWeb
OpenTV

table PlatformXSLFiles

PlatformID
XSLFileName

table AtomsCode

AtomID
XSLFileName
XSLCode

table ElementCode

ElementID
XSLFileName
XSLCode

table Function

FuncID
PlatformID
FunctionCode

table Classes

ClassName
PlatformID
BGColor
TextColor
FontFamily

FontSize
TextAlign
TextVerticalAlign
FontWeight
FontStyle
TextDecoration
Top
Left
Height
Width
Display
BorderWidth
BorderColor
TextShedow
Scroll

table AtomAttributs

AttributeID
AtomID

table ElementAttributs

AttributeID
ElementID

table FuncArguments

FuncID

ArgumentID
ArgumentType
argIADLSyntax

Fig. 34A – 34B, taken together, form a table summarizing the input parameters, output parameters and preferred mode of operation for wizard processing procedures useful in manipulating wizard data within the BackEngine database of Figs. 4 and 14.

Figs. 35A - 35B, taken together, form a table summarizing the input parameters, output parameters and preferred mode of operation for trigger processing procedures useful in manipulating trigger data within the BackEngine database of Figs. 4 and 14.

Fig. 36 is a table summarizing the input parameters, output parameters and preferred mode of operation for repository maintenance procedures useful in manipulating repository data within the BackEngine database of Figs. 4 and 14.

Fig. 37 is a table summarizing the input parameters, output parameters and preferred mode of operation for playlist processing procedures useful in manipulating playlist data within the BackEngine database of Figs. 4 and 14.

Fig. 38 is a table summarizing the input parameters, output parameters and preferred mode of operation for digital television trigger insertion procedures useful in manipulating data within the BackEngine database of Figs. 4 and 14 in accordance with the embodiments of Figs. 2 and 3.

Fig. 39 is a table summarizing the input parameters, output parameters and preferred mode of operation for packager procedures performed by the DTV packager of Figs. 4 and 12.

Fig. 40 is a table summarizing the input parameters, output parameters and preferred mode of operation for viewser log procedures useful in manipulating data within the BackEngine database of Figs. 4 and 14.

Fig. 41 is a table summarizing the input parameters, output parameters and preferred mode of operation for fusing procedures performed by the fuser of Figs. 4 and 9.

Fig. 42 is a table summarizing the input parameters, output parameters and preferred mode of operation for interactive serving procedures performed by the

interactive server of Figs. 4 and 10.

Fig. 43 is a table summarizing the input parameters, output parameters and preferred mode of operation for computer network trigger insertion procedures useful in manipulating data within the BackEngine database of Figs. 4 and 14 in accordance with the embodiments of Figs. 1 and 3.

Fig. 44 is a table summarizing the input parameters, output parameters and preferred mode of operation for procedures useful in manipulating application instance data within the application repository of Fig. 22.

Fig. 45 is a table summarizing the input parameters, output parameters and preferred mode of operation for application protocol interfacing procedures useful in manipulating data within the BackEngine database of Figs. 4 and 14.

Fig. 46 is a simplified pictorial illustration of a first screen display generated by the editor GUI (graphic user interface) of Fig. 4 when performing an example workflow.

Fig. 47 is a simplified pictorial illustration of a second screen display generated by the editor GUI (graphic user interface) of Fig. 4 when performing an example workflow.

Fig. 48 is a simplified pictorial illustration of a third screen display generated by the editor GUI (graphic user interface) of Fig. 4 when performing an example workflow.

Fig. 49 is a simplified pictorial illustration of a fourth screen display generated by the editor GUI (graphic user interface) of Fig. 4 when performing an example workflow.

Fig. 50 is a simplified pictorial illustration of a fifth screen display generated by the editor GUI (graphic user interface) of Fig. 4 when performing an example workflow.

A typical workflow for the Editor GUI 280 is now described.

For general usage of the Editor GUI 280: Editor-User launches the Editor-GUI 280 software application and defines the ePlaylist window 951's time range 952 (Fig. 46). The editor-user chooses the From/To date and time and clicks Go for submission.

The playlist 951 (Fig. 46) of the chosen date and time range appears,

containing the scheduled programs in the program column 954. Increasing or decreasing the level of detail (also termed herein "Zooming in and out") can be effected using the + and - zoom buttons 956. This action changes the number of min/sec each unit on the time column 955 represents.

The slider 959 allows scrolling along the selected time range and forwards/rewinds the video in the video preview window 969. The broadcast line 960 represents the current time and moves downwards as time passes.

The events column 953 presents the Triggers of interactive applications which were scheduled along the selected time range of the play list, each composed of two parts: an icon 957 which represents the type of the Trigger (e.g. confirmed, recurring, absolute, relative) and the name tag 958 (Fig. 48) of the application which is given by the Editor - User.

An example of how an Editor-User may form an Interactive Scheduled Application from an Interactive Application Template is now described:

The Editor User launches the Editor-GUI 280 software application, defines the ePlaylist window 951's time range 952, chooses the From/To date and time and clicks Go for submission. The Editor User selects a New template from the Template tab 964. The Editor User drags (as in "Drag and Drop" from Microsoft Windows Software) the selection onto an exact time along the Time column 955. The Application wizard (element 978 in Fig. 47) opens automatically.

The Editor User enters application general information and specifically he may Fill out the Name 985 (Fig. 48), Assigned element 979 (date and time that indicate beginning time of the event), and check the "specify duration" checkbox element 984 to limit the application duration (length of time it is available for the Viewser 180) and choose the duration time.

The Confirmed check box 983 is used for authentication purposes and privilege management. A relatively junior Editor-User can assign a non-confirmed application to the ePlaylist in Fig. 46 and a senior Editor-User can confirm it. Only confirmed instances are sent to the Broadcast Gateway 200 and 2001. The Absolute menu 980 (Fig. 48) enables the synchronization of Triggers to the ePlaylist 951 (Fig. 46) in a selectable one of the following two different modes:

1. Relative Time- Triggers that are attached to a program and

automatically adjusted to be broadcast at a predetermined time, according to the relative time within the actual program.

2. Absolute Time- Triggers that are assigned to the timeline, regardless of the program being broadcasted.

The Recurrence button (element 981 in figure 48) is enabled in both Absolute and Relative time modes.

Absolute Recurrence allows assignment of a Trigger to the Time line based on a recurrence pattern. The recurrence pattern may include seconds, minutes, hours, days and/or weeks. A range may be assigned as well, which can be one of three options: No end date; End after XXX occurrences; or End by DD/MM/YYYY (i.e. assigning a trigger every day at 16:00 that broadcasts a Promo for the Evening News.

Relative Recurrence Triggers may be attached to a program in a relative trigger mode (may be attached to a relative time within the actual program) and are typically automatically reassigned each time the program is re-broadcasted. Recurrent relative Triggers may be limited by a number of recurrences or by a range of dates.

Other Editor-User functions displayed in Figs. 46-50 include:

For entering an Interactive message: Editor User fills out the interactive message text 976 (Fig. 47), checks the Soft Launch checkbox 975 to determine if the application should initially appear on the TV screen as a small icon (clicking it invokes the interactive message), and leaves it unchecked if the application should send the full Interactive Application directly.

For Entering application relevant data fields: "step X of X" element 987 (Fig. 49) gives the Editor User an indication of the phases left for the completion of the editing process (i.e. step 1 of 1 in trivia application). The Editor User typically types the text for the question of the example - trivia (element 986 in Fig. 49), fills in the text of the possible answers and checks the radio button of the right answer.

For preview of a created Action or Instance: Click "Preview iTV" (element 988 in Fig. 50) to preview the application as it will be shown on the TV set. Click "Preview PC" (element 989 in Fig. 50) to preview the application as it will be shown on a PC screen.

For submitting an application to broadcast - Creating an Instance: In the application wizard 978 in Fig. 47, click Submit element 972 in Fig. 47) to save the

created Action in the BackEngine database 240 or if was assigned to the ePlaylist 951 in Fig. 46, or click Cancel (element 974 in Fig. 47) to exit the application wizard in which case typically all data entered will be lost.

For storing an application in the database - Creating a Action: Editor User double clicks an Application Template 963 (Fig. 46) in the Templates tab 964 (Fig. 46) and completes the authoring process in the same manner as described above. Saving it creates an Action which is saved in the BackEngine database 240 and displayed in the Action tab element 965 in Fig. 46 for future use.

An example of how an Editor-User may form an Interactive Scheduled Application from an existing Action stored on the BackEngine Database 240 is now described:

The Editor User launches Editor-GUI 280 software application and defines the ePlaylist window 951 (Fig. 46) time range 952, chooses the From/To date and time and clicks Go for submission.

The Editor User selects an existing Action from the Actions tab 965. The Editor User checks the Filters Box 968 tin to effect a refined search for specific Actions in the Actions tab 965. The View menu 967 in Fig. 46 enables the view of specific types of interactive applications in the actions tab 965 in Fig. 46.

The Editor User drags the selection into an exact time along the Time column 955 (Fig. 46). The Application wizard 951 automatically opens up in a partial mode.

The Editor User may edit the Name 985 (Fig. 48), Assigned at date and time that indicates beginning time of the application (element 979, and checks the Specify duration checkbox 984 to limit the application's duration and choose the duration time. A confirmed trigger 983 and an absolute trigger 980 are shown in Fig. 48 and recurrence is shown in element 981 in Fig. 48.

The rest of the process may be as described above, in the Interactive Application Template section.

An alternative way for the Editor-User to create and instance is now described. The Editor User may elect an alternative way of assigning a Template Interactive Application to time by placing the slider 959 (Fig. 46) at the desired time of synchronization on the ePlaylist timeline 951 and choosing Insert Event from the Events

menu 961. This invokes the opening of an application wizard 978 (Fig. 48. The Editor can then drag a new template from the templates tab 964 (Fig. 46) or an existing template from the Actions tab 965 and complete the editing process as described above. Once the editing process is completed, clicking Submit (element 972 in Fig. 50) assigns the application to the point indicated by the slider 959 (Fig. 46) along the ePlaylist timeline 955 (Fig. 46) and thus creates an instance in the BackEngine Database 240. the Editor-User creates an instance in real-time of broadcast (Online mode).

The Editor User chooses "Insert Immediate Event" from the Events menu 961 (Fig. 46). This invokes the opening of an application wizard 978 in Fig. 48. The Editor User then drags a new template from the templates tab 964 (Fig. 46) of the repository 968 or an existing template from the Actions tab 965 of the repository 968 and then completes the editing process as described above. Once the editing process is completed, clicking Submit (element 972 in Fig. 50) assigns the application to the point indicated by the slider 959 (Fig. 46) along the ePlaylist timeline 955 and thus create an instance in the BackEngine Database 240. The TIM server 210 receives a notification of an "immediate event" and sends the instance to the broadcasting process 1230 in Figure 53B.

Fig. 51 is a simplified flowchart illustration of a preferred method of operation for the system of Fig. 1.

Fig. 52 is a simplified flowchart illustration of a preferred method of operation for the system of Fig. 2.

Figs. 53A - 53B, taken together, form a simplified flowchart illustration of a preferred method of operation for the BackEngine of Figs. 1, 2 and 4.

Knowledgebase 250 of Fig. 4 is used by DTV Packaging Subsystem 270 to translate interactive template applications from IADL into target platform code. Conventionally, external application providers particularly in DTV environments such as an OpenTV environment, provide applications in a target platform code i.e. code understandable by a target platform such as the external broadcast operator's data carousel of Fig. 2. According to a preferred embodiment of the present invention, interactive template applications may be received from an external applications provider in target platform code, are typically wrapped, by fuser 110 of Fig. 2, in an IADL shell for processing by the BackEngine 100.

Fig. 54 is a diagram showing a typical life-cycle of a modular interactive application in accordance with a preferred embodiment of the present invention.

Fig. 55 is a simplified flowchart illustration of a preferred method by which application composer 170 of Figs. 1 and 6 performs the interactive application file generation step 1010 of Fig. 51;

Fig. 56 is a simplified flowchart illustration of a preferred method by which fuser 110 of Figs. 1 and 9 performs the interactive application template generation step 1030 of Fig. 51.

Figs. 57A - 57B, taken together, form an example of a main.XML file that describes the syntax of any interactive application file generated by the application composer 170.

Fig. 58 is a simplified flowchart illustration of a preferred method whereby the editor GUI 280 in BackEngine 100 of Figs. 1 and 4 perform a first part of the interactive scheduled application generation step 1040 of Fig. 51, corresponding to content injection step 1190 in Fig. 53A.

Fig. 59 is a simplified flowchart illustration of a preferred method whereby the editor GUI 280 in BackEngine 100 of Figs. 1 and 4 perform a second part of the interactive scheduled application generation step 1040 of Fig. 51, corresponding to assignment to timeline step 1210 in Fig. 53A.

Fig. 60 is a simplified flowchart illustration of a preferred method whereby the BackEngine of Figs. 1 and 4 performs the interactive scheduled application step 1044 of Fig. 51.

Fig. 61 is a simplified flowchart illustration of a preferred method whereby the BackEngine 100 of Figs. 1 and 4 performs the on-air signal receiving step 1046 of Fig. 51.

Fig. 62 is a simplified flowchart illustration of a preferred method whereby the IP broadcast gateway 201 of Figs. 1 and 7 performs the interactive scheduled application broadcasting step 1050 of Fig. 51.

Fig. 63 is a simplified flowchart illustration of a preferred method whereby the viewser 180 uses his PC to generate a viewser response which is subsequently processed by the system of the present invention in steps 1060 and 1070 of Fig. 51. The interface described by the flowchart of Figs. 62 and 63 operates in a

suitable environment which typically includes the following components: a browser such as Netscape Explorer, a media player such as Windows Media Player, an operating system such as Windows XP, and a suitable communication driver to the network such as the 3Com Type M Modem.

Referring again to Fig. 2, the destination of the sequence of scheduled interactive applications is a population of television systems including a conventional television set, a conventional set-top box such as a Digibox set-top box marketed by Pace Micro Technology, Victoria Road, Saltaire, Shipley, West Yorkshire BD183LF, UK, and suitable middleware running in the set-top box, such as OpenTV middleware marketed by OPENTV Corp. 401 East Middlefield Road, Mountain View CA 94043, USA. Each scheduled interactive application arriving at each television system's set-top box typically includes a "thin-client" wrapper program wrapped around interactive application logic. Preferably, the sequence of scheduled interactive applications is forwarded to the population of television systems via an external broadcast operator equipped with a suitable forwarding mechanism such as data carousel software e.g. OpenStreamer marketed by OpenTV. When the scheduled interactive application arrives at the set-top box the set-top box typically initially interacts with the "thin-client" wrapper program rather than with the interactive application logic. The "thin-client" wrapper program activates the interactive application and manages its communication with its environment as described in Fig. 11.

Fig. 64 is a simplified flowchart illustration of a preferred method whereby the BackEngine 100 of Figs. 1 and 4 performs the viewser response processing step 1070 of Fig. 51.

Fig. 65 is a simplified flowchart illustration of a preferred method whereby the feedback system 160 of Figs. 1 and 12 performs the viewser response statistics reporting step 1080 of Fig. 51.

Fig. 66A is a simplified flowchart illustration of a preferred method whereby the BackEngine of Figs. 2 and 4 performs the interactive scheduled application sending step 1124 of Fig. 52.

Figs. 66B - 66C, taken together, form a simplified flowchart illustration of a preferred method whereby the DTV packaging subsystem 270 of Figs. 4 and 13 performs the DTV packaging step in the method of Fig. 66A, thereby to generate a

packaged instance.

Fig. 66D is a simplified flowchart illustration of a preferred method of operation for the IADL transformer 680 of Fig. 13.

Fig. 67 is a simplified flowchart illustration of a preferred method whereby the DTV broadcast gateway 200 of Figs. 2 and 8 performs the interactive scheduled application broadcasting step 1130 of Fig. 51.

Fig. 68 is a simplified flowchart illustration of a preferred method whereby the DTV broadcast gateway 200 of Figs. 2 and 8 performs the viewser response receiving step 1140 of Fig. 52.

Fig. 69 is a simplified flowchart illustration of a preferred method whereby the viewser uses interactive application digital TV interface software typically residing within his set-top box according to a preferred embodiment of the present invention, to generate a viewser response which is subsequently processed by the system of the present invention in steps 1140 and 1150 of Fig. 52.

Fig. 70 is a simplified flowchart illustration of a preferred method whereby the sync driver 220 of Fig. 4 performs the playlist processing step of Fig. 53A whereby the playlist is prepared for display by GUI 280 of Fig. 4.

Figs. 71A - 71B, taken together, form a simplified flowchart illustration of a preferred method whereby the trigger insertion mechanism server 210 of Fig. 4 performs the interactive scheduled application generation step of Fig. 53B.

A preferred language for defining interactive application templates is now described with reference to Figs. 72A - 95B. This language or core syntax is termed herein IADL (Interactive Application Definition Language). IADL may be used for defining the logic and behavior of actions in the system of the present invention. It is a syntax that may be used to commonly describe the business logic of applications, thus enabling multi-platform display.

IADL preferably comprises a flexible development environment for the creation of interactive applications. IADL is typically targeted at network operators, multi service operators (MSO's) and independent application developers. IADL preferably provides ease of use and platform portability. IADL preferably provides the developer with familiar and easy to use building blocks called Elements and a Web like development environment. IADL typically operates in accordance with a CODE (Create

Once Display Everywhere) mode.

IADL is typically based on the following four logic layers:

a. Atoms: basic building blocks. Examples of Atoms: "text", "image", "sound".

b. Elements: A group of Atoms with a defined functionality forms an Element - an easy to use object that is designed to execute a familiar functionality. Examples of Elements: "running text", "list".

c. Stages: Screens for TV which define the appearance of elements on screens and functionality for screen level. Example: "stage" detects contradictions in use of the digital TV's remote control buttons.

d. Application: Contains the flow (logic) of the screens and functionality in application level. Example: The function may know what Elements can be reused in the application in order to save bandwidth and memory

Fig. 72A is a table describing two IADL application-level commands having a common syntax described in Fig. 72B.

Fig. 73A is a table describing four IADL stage-level commands having a common syntax described in Fig. 73B.

Figs. 74A, 75A, 76A, 77A, 78A, 78A, 79A, 80A, 81A, 82A, 83A, 84A, 85A and 86A are tables, each row of which describes an element-level IADL command wherein the commands in each such table have a common syntax as described in Figs. 74B, 75B, 76B, 77B, 78B, 78B, 79B, 80B, 81B, 82B, 83B, 84B, 85B and 86B respectively. and

Figs. 87A, 88A, 89A, 90A, 91A, 92A, 93A, 94A and 95A are tables, each row of which describes an atom-level IADL command wherein the commands in each such table have a common syntax as described in Figs. 87B, 88B, 89B, 90B, 91B, 92B, 93B, 94B and 95B respectively.

Fig. 96 is a simplified pictorial illustration of a first screen display generated by the Application Builder 320 and its GUI 330 (Fig. 6) when performing an example workflow as described in Fig. 55. The Application Builder is a preferred tool for the creation of Interactive Application templates.

Figs. 97A - 97F are simplified pictorial illustrations of the stage(skeleton), project browser, elements inspector, saved elements, functions and

atoms library windows, respectively, in the screen display of Fig. 96.

Forming part of a set of tools preferably provided in accordance with the present invention, Application Builder 320 and its GUI 330 perform an initial portion of the application creation process. Once an application template has been created by the Application Builder 320 and its GUI 330, the template may be stored in the BackEngine Database 240 and may be populated with information by the editor-user using the Editor-GUI 280.

The Application Builder 320 is preferably operative to selectably open, save and modify Application Builder 320 projects. An Application Builder 320 project can be published to an Editor Suite. A published application typically comprises the following components: the application code, typically targeted to a specified iTV platform, one or more application skins, and a population Wizard, for use in an Editor Suite. The application is typically defined on paper by its author (briefing), after which both the programmer and the designer work on it together. The programmer is typically first to use the application builder, creating a working mock-up of the application (skeleton). In parallel, the designer works with external design tools such as Adobe's Photoshop, on the application design. After the skeleton has been created, the designer typically imports the graphic, and builds the application's first (or single) skin. Preferably this interactive authoring flow is a two-way-flow.

One application may contain several, different, skins. Since the process of the skeleton creation may be complex, including debugging and testing, the client may wish to re-use application skeletons as much as they can, and adapt them to different TV shows. For example, a survey about an item from the evening news could easily become a survey about an item from a sports broadcast because although the two applications might look entirely different, they may share the same logic. The system give the designer the ability to create new skins for an existing application without the need to communicate directly with the programmer who created the application.

Therefore, the "skeleton" and the "skin" are typically separated into two different, independent entities. This allows maximum freedom in the creation process, i.e. a designer can pickup an existing skeleton and create a new skin for it without any assistance from the programmer, and a programmer can create a new application using only the default skin. However, a skin can typically be applied only to the application it

has been created for.

The Application Builder 320 automatically discerns the distinction between Skeleton and Skin elements. For instance, if an author imports a custom library that contains both visual elements (pictures) and logic elements (functions), The application builder can display each one of their properties in a relevant tab and save them in the relevant entity. In case the skeleton has been locked, logic object placements are typically forbidden.

It is appreciated that according to a preferred embodiment of the present invention, the modular interactive application generation system shown and described herein in Figs. 1 and 2 and onward is operative to facilitate generation of interactive applications which are modular in at least one and preferably all of the following senses:

a. At least some interactive applications initially exist as a template which is modular in the sense that any suitable interactive content can be injected into the template to generate a plurality of different content-specific interactive applications from a single modular content-independent interactive application predecessor.

b. At least some interactive applications exist in a time independent form, termed herein "action" form, which is modular in the sense that each such action can be associated with any of a plurality of time-points along a timeline, to generate a plurality of different time-synchronized interactive applications, termed herein "instances", from a single modular time-independent interactive application predecessor;

c. At least some interactive applications exist in platform-agnostic form, such as "IADL format" shown and described herein, which is modular in the sense that each such platform-agnostic interactive application can be platform-specifically packaged to establish compatibility with a plurality of different interactive broadcasting platforms (such as OpenTV, MediaHighway, and Liberate). Thereby, a plurality of different platform-specific interactive applications are generated from a single modular platform-agnostic interactive application predecessor.

The database manager associated with BackEngine database 240 in Fig. 4 is operative to perform a variety of interactive application data processing operations also termed herein "database stored procedures". In the illustrated embodiment, these are implemented as Oracle "stored procedures".

It is appreciated that at least three modes of associating interactive items with a timeline are afforded by the system of the present invention: off-line editing mode, immediate-broadcast editing mode and real-time editing mode. The off-line editing mode is used for pre-recorded programs such as commercial programs, the system preferably allows the editor-user to view the interactive item he has generated and temporally associated with the playlist, in conjunction with the concurrently running program.

IADL (interactive application definition language) is an example of a suitable language for writing template applications for wizard editing. A client can generate his own template using the IADL language described herein or any other suitable interactive application definition language.

A content editor or editor may generate content using the system of the present invention. This content is automatically processed by the system in order to adapt it for viewing on a television screen or computer screen or small-screen display devices such as cellular communication devices or any display device whose characteristics are known to the system.

The TV knowledgebase of the present invention is preferably read by a transformer inside the broadcast gateway which converts IADL to a target device-understandable language.

An example process of planning, creating, editing, broadcasting and monitoring the results of Survey-type Interactive Applications, is now described. The survey-type interactive applications are to be presented to a plurality of end-users, at a precise assigned timing reflecting context-linking to the video content that is broadcast live or broadcast off-tape. Context linking is typically performed by a human operator termed the editor-user, in conjunction with the editor GUI 280 (Fig. 46) shown and described herein. In off-tape broadcasting, context linking is typically performed off-line whereas in live broadcasting, context linking is performed on-line.

In this example, the end-users receive and interact with the context-linked interactive application on two of a plurality of end-user devices encompassed within the scope of the present invention: A Television set running within a digital TV network, and a Personal Computer running within an Internet Protocol Network.

The entities involved in the process typically comprise a broadcaster, an

operator and viewsers as described in detail herein.

Channel 1 (The Broadcaster): A content packaging entity responsible for broadcasting a sequence of programs, typically video programs, including editorial programs and advertisement programs. Persons involved in the broadcasting process, in Channel1, may include:

Editor-in-Chief: Makes final interactive and video content decisions

Interactive Designer: Designs the look and graphics ("skin", i.e. visual wrap) of the interactive application.

Interactive Programmer: programs the interactive application template typically using the application composer 170 and IADL (interactive application definition language whose syntax is represented in Figs. 72 - 95) shown and described herein.

Editor-User: Enters changeable content data to the Interactive Application Template and links the resulting interactive application to a specific time-point in the video program broadcast schedule.

WebJockey Editor-User: Usually operates in Channel 1's control room. Monitors the system described herein using the editor GUI 280. Able to inject interactive applications into a video program stream using the editor GUI, particularly if the video program stream is being aired live.

CableSatCo (the Operator): An entity engaged in the business of providing a network that delivers televised content including interactive content to a plurality of end-users. Usually hosts the broadcasting mechanism ("headend") for both the video and the interactive programming. In this example CableSatCo. provides the content on both the digital television network and the broadband Internet network. In case of Digital TV the Operator usually provides each viewser in the viewser population it serves with a Set-Top Box which typically comprises a compatible device able to retrieve from a received videostream, and to display on the viewser's TV set, the video and interactive content sent by the operator. The Set-Top box and the headend are equipped with middleware and data broadcasting software and hardware using suitable matching technologies such as OpenTV.

PC Viewser (Jill): An end-user using her Personal Computer, featuring an Internet connection, a web browser (such as Microsoft Internet Explorer) and a

Media Player software (such as Microsoft Windows Media Player) to view and interact with the content provided by the CableSatCo.

DTV Viewser (Jack): An end-user using his Digital TV set, Set-Top Box and remote control to view and interact with the content provided by the CableSatCo.

An example of a sequence of events culminating in production of a video sequence including interactive elements in accordance with a preferred embodiment of the present invention is now described:

Monday, 9:00 AM: The Editor-in-Chief at Channel 1 prepares a creative brief (step 1400 in Fig. 55) typically comprising a single page which may specify that three interactive applications should be prepared and synchronized to Wednesday evening's 7 PM - 9 PM programming slot, and may further specify that the interactive application should include the following three elements:

a. an off-line "musician" survey allowing viewsers to select their favorite musician, to be aired during a teenager show featuring music video clips- scheduled to air Wednesday 7:00 PM. The teenager show is to be followed, in the schedule by an open slot at 7:45 PM in which a song by the musician favored by the largest number of viewsers will be aired.

b. an off-line commercial "car" survey encouraging viewsers to select their favorite car and thereby become eligible to participate in a lottery, the commercial survey to be aired in the course of a car commercial to be aired during the teenager show, at 7:30:30 PM.

c. a live newscast showing a political debate from the parliament starting at 8:00 PM, the debate to be overlaid with an on-line "political" survey question to be determined on-line by the web jockey editor-user as a function of the content of the live debate.

The two off-line Survey applications may be prepared and assigned to the ePlaylist 953 (Fig. 46) by the Editor-user 24 hours prior to the scheduled broadcast. The on-line "political" survey application may be prepared in the first few minutes after the show begins, and is typically ready to broadcast approximately 5 minutes later. Once the on-line survey application is ready to broadcast, the WebJockey Editor-User may insert the application into broadcast, in real-time, according to the events occurring in the live broadcast of the news-cast - in order to create an impression of context-

relevancy.

Monday 10:00 AM: The Interactive designer and the programmer receive the creative brief and prepare a application logic tree (step 1410, Fig. 55) for use as basis for all three of the planned surveys. The logic tree contains a field for a survey question, a plurality of possible answer fields e.g. 4 answer fields, and a "see results" button for immediate generation of current results. The designer sets off to create six sets of graphic files ("skins"), a file for PC viewing of each of the three surveys and a file for DTV viewing of each of the three surveys. The programmer engages in constructing the "skeleton" or logic of the survey application using the application builder 320 (Figs. 96 and 97), and generating code such as the one described in Figs. 57A - 57B. The "skeleton" can be used for all three surveys, both for PC and DTV viewsers.

Monday 5:00 PM: The programmer receives the graphic files from the designer, and after approval of the Editor-in-Chief, assembles and creates three templates for the three surveys respectively (step 1440, Fig. 55). The three templates preferably leave a degree of freedom in content selection such that they are somewhat flexible. Alternatively, a single template may be generated in which case, preferably, the skin is selected from within the template. The programmer also creates a wizard (step 1450, Fig. 55) which, once attached to the logic, allows the editor to incorporate desired content into the specific scheduled interactive application.

Monday 6:00 PM: The programmer previews the application and tests it with each of the six "skins" prepared by the designer (step 1470, Fig. 55.) The programmer then exports (step 1500, Fig. 55) the application to the Fuser 110 (Fig. 9). The fuser automatically stores the template application and its relevant "skins" in the BackEngine Database 240.

Tuesday 9:00 AM: The Editor-User reviews the creative brief and begins an editing session at the Editor-GUI 280 workstation which typically includes the following operations:

The Editor User points the displayed schedule 952 (Fig. 46) to Wednesday between 7:00 and 9:00 PM, and the planned playlist is automatically displayed in the ePlaylist window 954 (Figs. 46 and 70). The Editor User Opens the repository 962 (Fig. 46) and searches using the view selection box 967 (Fig. 46) in the

repository window, for the musician Survey application.

The Editor-User then drags the musician survey template to 30 seconds past the planned beginning of the broadcast, i.e. 7:00:30 PM. The point dragged to generates a visible trigger (elements 957 and 958 in Fig. 46) and a wizard window pops up (Figure 47) and allows the editor to enter the following survey content data in accordance with the process described in Figs. 58 and 59: a suitable opener such as "select the performer of the day", names and pictures of 4 performers, and additional information on their latest release.

The Editor-user previews the application, and sends a query to the feedback module 230 (Fig. 64) to display the results of this survey at 7:15 PM at the WebJockey Editor-GUI workstation. The Editor-User then submits the interaction (step 1210, Fig. 53A), automatically registering an instance in the BackEngine Database 240 (Figure 60.)

The Editor User then repeats the above steps, starting from template dragging, however this time, the template is dragged to a different temporal location e.g.7:35:00 PM, thereby to generate an additional survey, this time titled "select the worst artist of the day", from the same musician survey template.

Tuesday 11:00 AM: The Editor User receives content and graphic materials from an advertising agency whose client is the Compact Car manufacturer. These materials are to be used for a commercial to be aired at exactly 07:30:30 PM on Wednesday. The Editor-User then enters the data to the car survey template application, and uses a suitable survey question such as : "which car is the best value for your money -- enter your choice and you may win a prize!". A query is generated and forwarded to the feedback system 160 (Figs. 12 and 65) -asking the feedback system to forward to the advertiser a report, e.g. a named report, of all viewsers that have elected to answer the survey.

The editor-user assigns the Action (interactive application with content) to the beginning of the commercial in the ePlaylist, and preferably specifies a duration e.g. of 1 minute (element 984, Fig. 48) for the display of the question to the viewsers, after which the question disappears from the screen.

According to the placement agreement between the advertising agency and Channel 1, this commercial is planned to be aired 24 times during the following

week, so the Editor User specifies in the recurrence window (element 981 in Fig. 48) that each time the commercial airs, the interactive application will be aired as well.

Tuesday 6:00 PM: The Editor-in-Chief previews (elements 988, 989 in Fig. 50) all applications created by the Editor-User (in this case, the two musician surveys ("best" and "worst", and the car survey) and confirms, using confirm button 983 (Fig. 48), the "best musician" and "car" surveys but rejects the "worst musician" survey. The PC applications are now stored in the Interactive Server 150 (Figs. 10 and 60), ready to be sent. The DTV applications are now stored as instances with reference to all resource files, waiting to be packaged and compiled.

Wednesday 6:30 PM: The TIM Server 210 receives an instruction to send the approved instances i.e. in the present example the "best musician" and "car" survey instances, to the DTV Packaging Subsystem (Figs. 13 and 66A), where both survey instances are compiled (Figs. 66B - 66D) and sent via a TCP/IP dedicated line to the DTV broadcast Gateway 200 residing at CableSatCo's Headend location. The two DTV Interactive Scheduled Applications are then sent to the CableSatCo's data carousel 203 (Figs. 2 and 67).

Wednesday 7:00 PM: An on-air signal is received by the system for the teenage show (Figs. 61 and 62 for IP to PC and Fig. 67 for DTV.) The on-air signal shows that there has been a delay of 1 second compared to the planned time of broadcast of the teenage show. The TIM Engine (element 360, Fig. 7 and element 400, Fig. 8) then computes this delay and sends the trigger command (Figs. 71A - 71B) one second later than the planned airing time defined in the clock (element 365 in figure 7 and 410 in Fig. 8).

Wednesday 07:00:31 PM: Jill and Jack, the PC viewser and the DTV viewser, both watching the video programming, receive an interactive message and an application loader respectively. They both decide to interact (Figs. 63 and 68-69 for PC and DTV respectively.

Jill, the PC viewser, clicks with the mouse on the interactive message, activating an IP session with the survey application residing at the Interactive Server 150. She then answers the question and sends her input to the system.

Jack, the DTV Viewser clicks a Red button on the remote control (confirming he wanted to participate in the survey) and uses the arrows and select keys

in the remote control to vote. His input is registered in CableSatCo's network and sent to the Return Channel server 260 (see Fig. 68) and to the Feedback Module 230 of the system.

Wednesday 07:15:00 PM: Jack and Jill's (and other viewsers') inputs are presented in the report generated by the feedback module. The Video Producer in charge of the control room, loads the relevant winning piece from the video server and airs this democratic selection made by the people.

Wednesday 07:30:30 PM: The "car" survey application is aired exactly as the commercial begins, despite a delay of 10 seconds in the planned time. This is because the On-air signal is responsible to notify the entire system (Figs. 71A - 71B) for this delay, ensuring precise synchronization of the display of the interactive message (IP) and the application loader (DTV) to the start time of the commercial.

Jack and Jill's clicks to answer the survey result are registered in the Feedback Module and a report is generated for the advertiser.

Wednesday 08:00:00 PM: The WebJockey Editor-User is following the live broadcast of the parliamentary debate. It turns out that the topic of the debate is the country's anti-racism policy. Joe, a speaker representing an extremist party requests permission to make a sensational statement. While the house debates whether to let him speak, the WebJockey Editor User, anticipating that Joe will be allowed to speak, prepares a quick action using the political Survey template application, using the following hastily worded question: "Do you support Joe's statement?". The statement itself, of course, has not yet been made and therefore has not yet been aired. The WebJockey Editor User also preferably fills in 4 possible answers: "Very Much So!" "Somewhat..." "Disagree" and "No opinion". Alternatively, the answers may have been predetermined in the wizard. The application content generated by the WebJockey Editor User is saved as an action in the BackEngine Database and represented in the Actions tab (element 965, Fig. 46) in the repository window.

Wednesday 08:07:25 PM: As soon as Joe rises to begin his statement, the WebJockey Editor-User chooses "insert immediate event" from the Events menu (element 961 in Fig. 46), and activates submission of the action for broadcast, thereby creating an instance for immediate delivery to viewsers. This creates an impression of a real-time response to events occurring in the video feed. Preferably, the wizard logic

dictates that the survey results are immediately displayed to all viewsers.

It is appreciated that the software components of the present invention may, if desired, be implemented in ROM (read-only memory) form. The software components may, generally, be implemented in hardware, if desired, using conventional techniques.

It is appreciated that various features of the invention which are, for clarity, described in the contexts of separate embodiments may also be provided in combination in a single embodiment. Conversely, various features of the invention which are, for brevity, described in the context of a single embodiment may also be provided separately or in any suitable subcombination.

It will be appreciated by persons skilled in the art that the present invention is not limited to what has been particularly shown and described hereinabove. Rather, the scope of the present invention is defined only by the claims that follow:

CLAIMS

1. A system for generating interactive television programs comprising:

an interactive item scheduler operative to generate an interactive item schedule for incorporation into at least one television program, the interactive item schedule comprising a first plurality of interactive items each associated with a time-stamp; and

an interactive television program integrator operative to incorporate said first plurality of interactive items into at least one television program in accordance with said schedule.

2. A system according to claim 1 wherein said interactive television program integrator is operative to receive, for each individual one of at least one television programs, an on-air signal indicating, in real-time, the time at which broadcast of the individual television program began.

3. A system according to claim 1 wherein said interactive television program integrator is also operative to receive, in advance of broadcast, from an external source, a playlist comprising a second plurality of television programs to be broadcast and to generate, off-line, an output instruction to a broadcasting facility describing how to incorporate said first plurality of interactive items into said second plurality of television programs in accordance with said schedule.

4. A system according to claim 3 and also comprising an interactive television GUI operative to generate a graphic display of the playlist and of a library of interactive items and to accept an editor-user's input associating an individual interactive item from the library with a temporal location on the playlist.

5. A system according to claim 4 wherein the graphic display also comprises a video window which, responsive to a user's indication of a temporal location on the playlist, presents a portion of a program associated with said temporal

location.

6. A system according to claim 5 wherein the video window, responsive to an editor-user's input associating an individual interactive item from the library with a temporal location on the playlist, presents a portion of a program associated with said temporal location and, concurrently, the portion of the individual interactive item associated with said temporal location.

7. A system according to claim 1 wherein said interactive television program integrator is operative to display said first plurality of interactive items concurrently with a corresponding first plurality of portions of at least one television program in accordance with said schedule.

8. A system according to claim 7 wherein said interactive television program integrator is operative to superimpose at least one of said first plurality of interactive items onto at least one of the corresponding first plurality of portions of at least one television program in accordance with said schedule.

9. A system according to claim 1 wherein said interactive item scheduler comprises an interactive item generator operative to generate at least one interactive item for inclusion in the interactive item schedule.

10. A system according to claim 9 wherein said interactive item generator comprises a library of empty interactive item templates and a template filling user interface operative to accept, from an editor-user, interactive content to fill an editor-user-selected one of the interactive item templates.

11. A system according to claim 10 and also comprising a repository for filled interactive item templates thereby to enable an editor-user to fill templates off-line for real time incorporation into at least one television program.

12. A system according to claim 1 wherein at least one time-stamp for at

least one individual interactive item comprises an absolute time for broadcast of the individual interactive item.

13. A system according to claim 1 wherein at least one time-stamp for at least one individual interactive item comprises a time for broadcast of the individual interactive item, relative to an on-air signal to be received which will indicate the time at which broadcast of an individual television program began.

14. A methodology for providing enhanced television type content to a plurality of disparate displays comprising:

providing television type content;

enhancing said television type content in a display-independent manner to provide enhanced display-independent interactive television type content; and

providing a plurality of display specific additions to said enhanced display-independent television type content.

15. A methodology for providing enhanced television type content to a plurality of disparate displays according to claim 14 and also comprising broadcasting said enhanced display-independent television type content with at least one display specific addition.

16. A methodology for providing enhanced television type content to a plurality of disparate displays according to claim 15 and also comprising:

receiving and displaying, at a given one of said plurality of disparate displays, said enhanced display-independent television type content with at least one display specific addition.

17. A system for authoring and broadcasting of interactive content, the system comprising:

creation of interactive content by non-programmers including at least one of the following editing functions:

drag-and-drop function for incorporation of interactive content into a

program schedule;

wizard-based content creation for interactive content; and

editing-level synchronization with broadcasting events including a synchronization information display for the non-programmer interactive content creator.

18. Interactive content screen display apparatus comprising:

a first video area portion displaying a video broadcast;

a second interactive portion displaying interactive content selected by a viewer; and

a third pushed interrupt portion, which cannot be overridden by the viewer, displaying interrupting interactive content pushed by an interactive content provider,

and wherein the second interactive portion cannot be overridden by the interactive content provider.

19. A system for conveying interactive content to a plurality of user terminals having different characteristics, the system comprising:

an interactive content generator; and

a plurality of user-terminal specific compilers operative to compile interactive content generated by the interactive content generator so as to adapt said interactive content for use by a corresponding one of said user terminals,

thereby to provide interactive content generated by said interactive content generator to all of said plurality of user terminals despite their different characteristics.

20. A system according to claim 19 wherein the user terminals differ with respect to at least one of the following types of terminal characteristics:

user terminal operating system characteristics

user terminal output characteristics

user terminal input characteristics.

21. A system according to claim 19 wherein said interactive content

generator comprises:

a library of templates, each template being operative to prompt a content editor to fill the template with specific content, thereby to generate a template instance comprising an action.

22. A system according to claim 21 wherein each template is operative to prompt the content editor to define a template instance trigger thereby to generate an assigned action.

23. An interactive content generation system comprising:

an interactive content template repository storing a plurality of templates for interactive content items; and

a template filling interface allowing a user to select, view and fill in a template from among said plurality of templates.

Figure 1

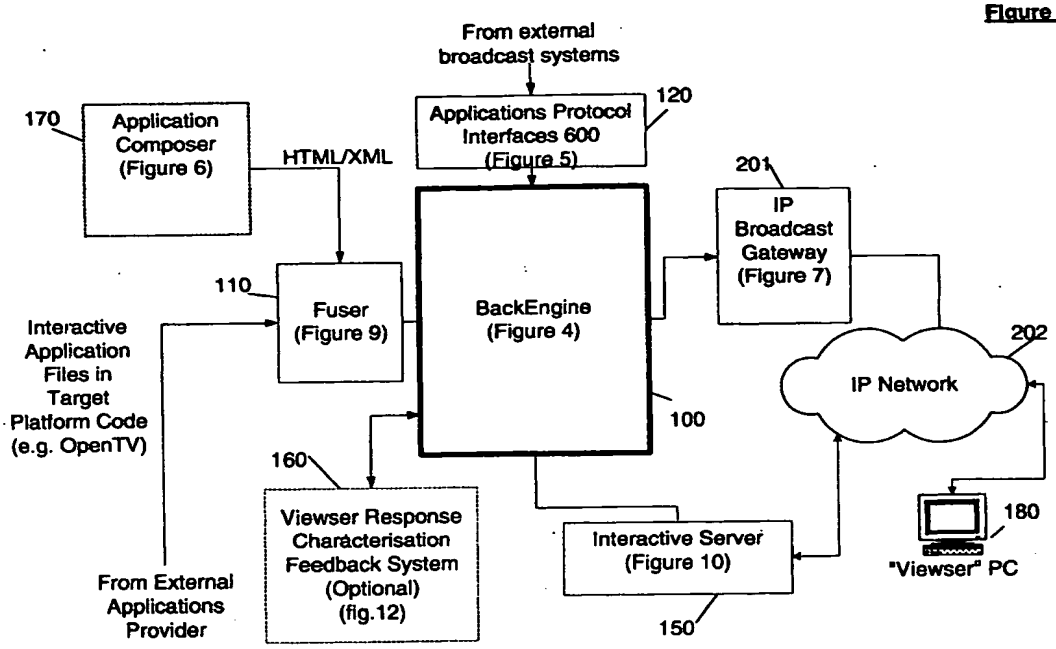


Figure 2

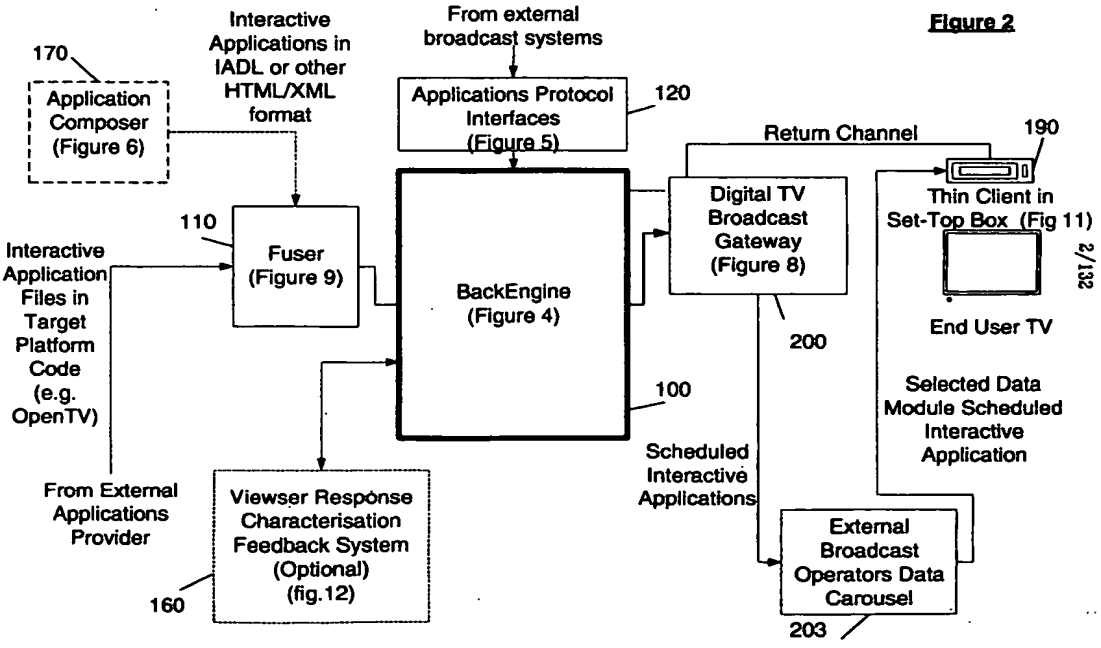


Figure 3

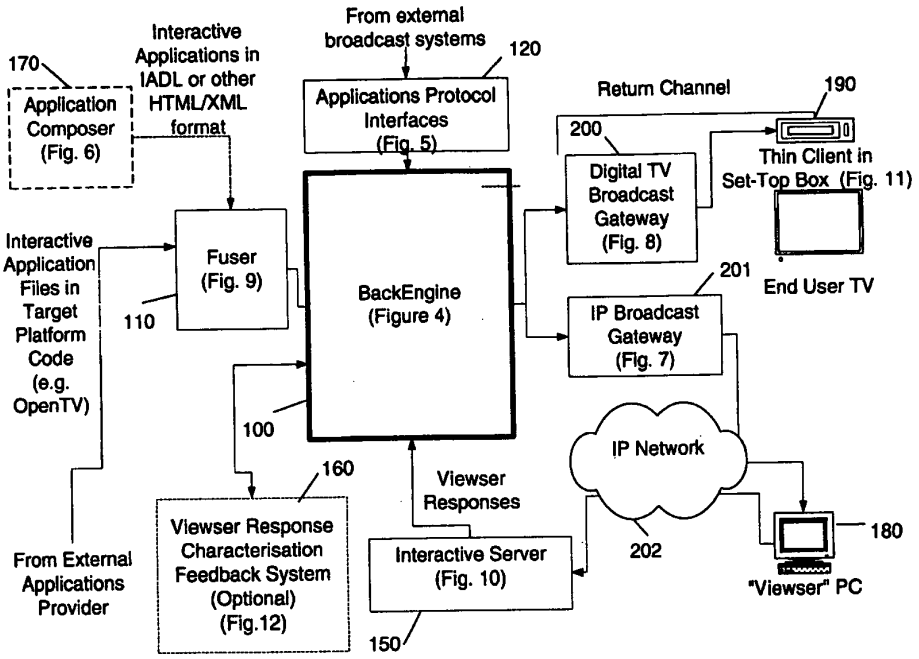


Figure 4

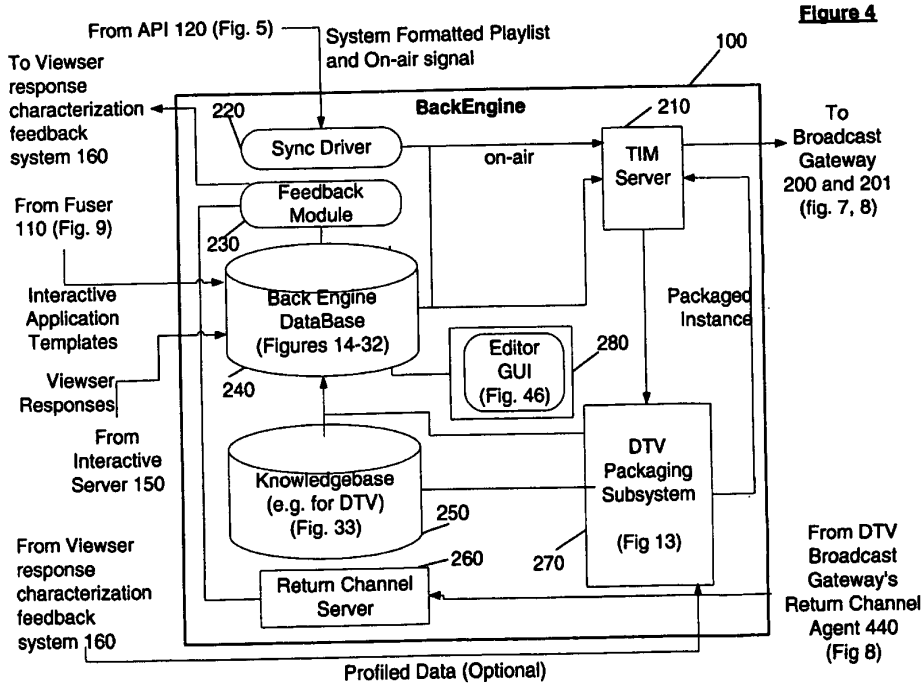


Figure 5

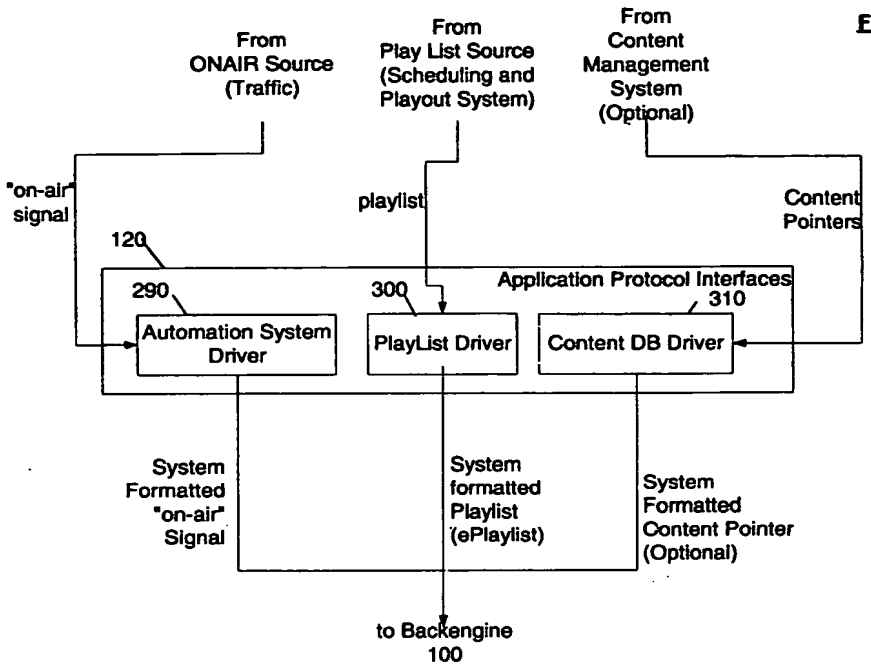


Figure 6

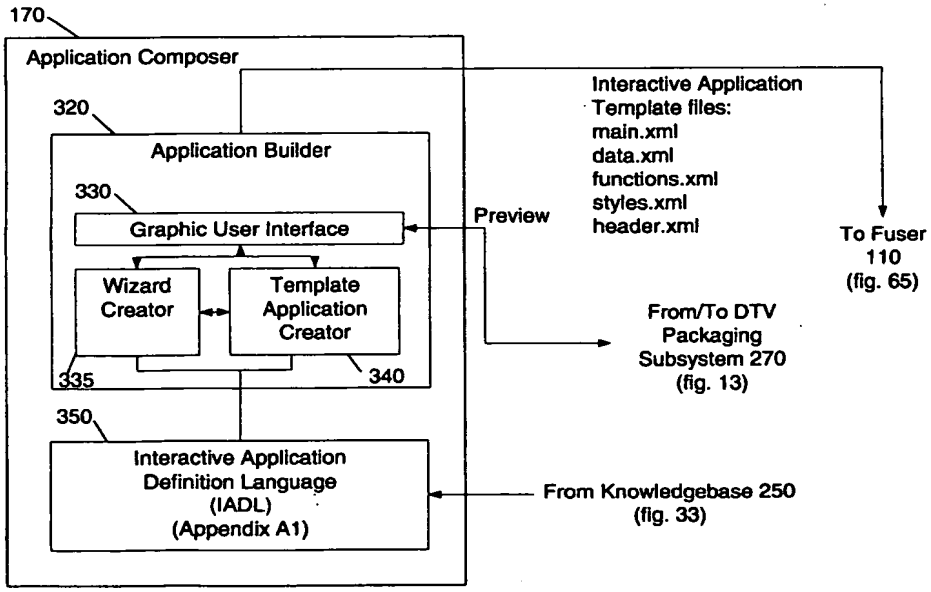


Figure 7

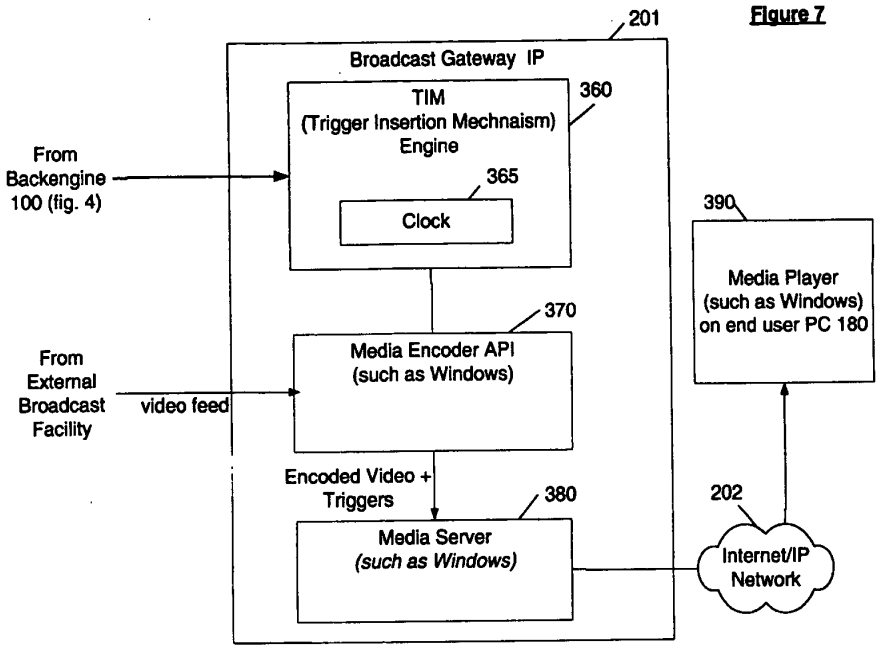


Figure 8

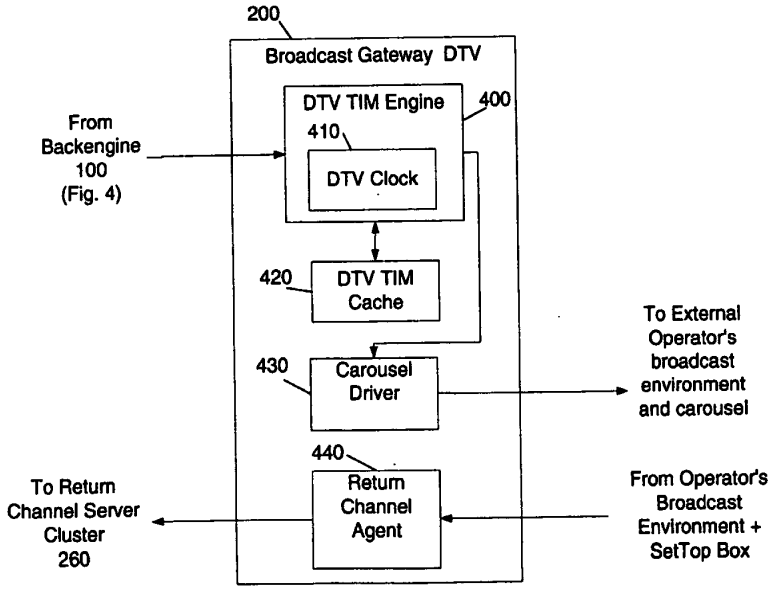


Figure 9

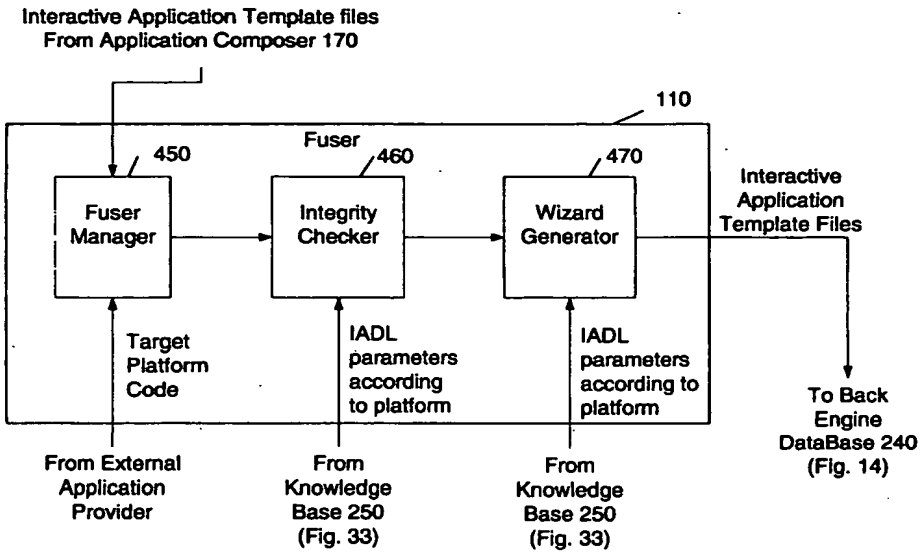


Figure 10

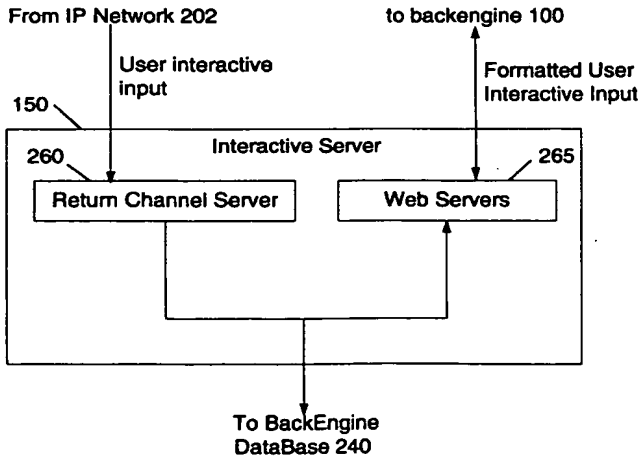


Figure 11

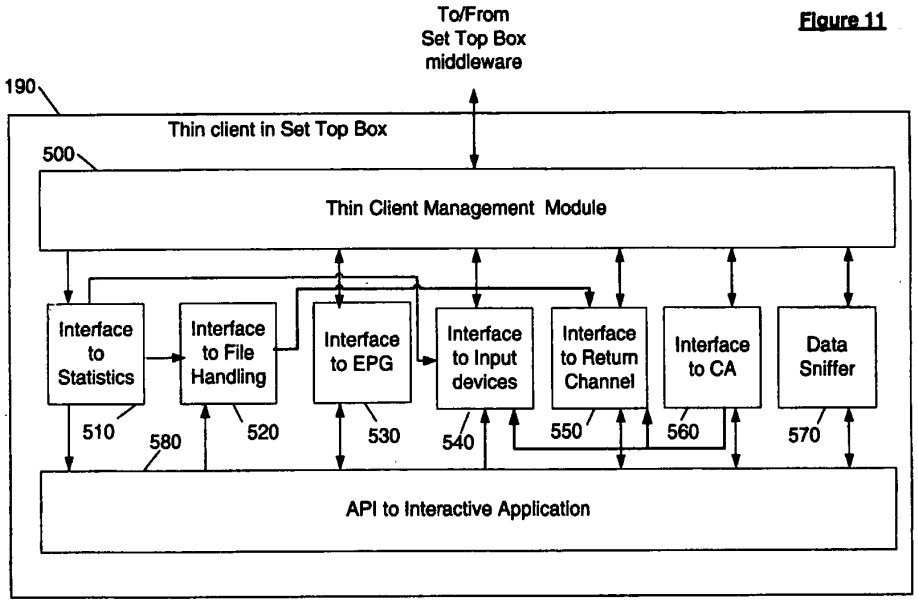
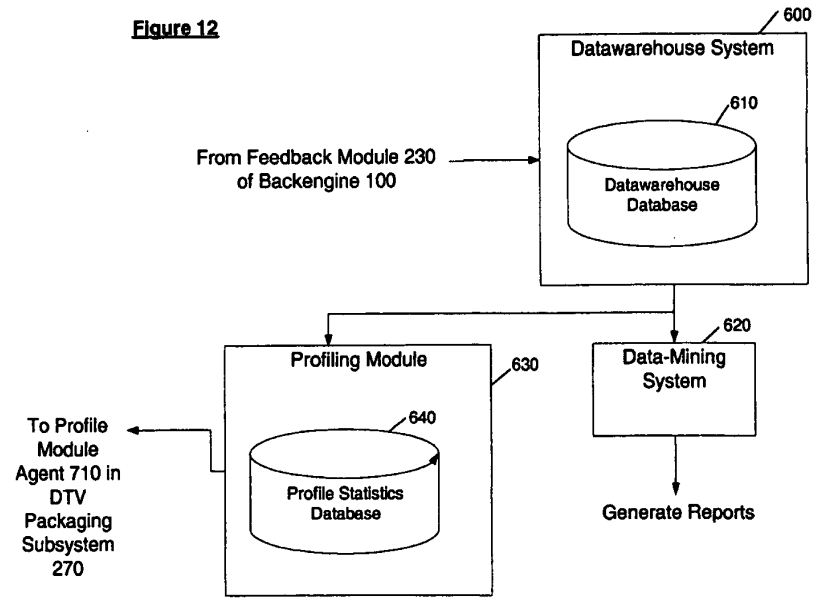


Figure 12



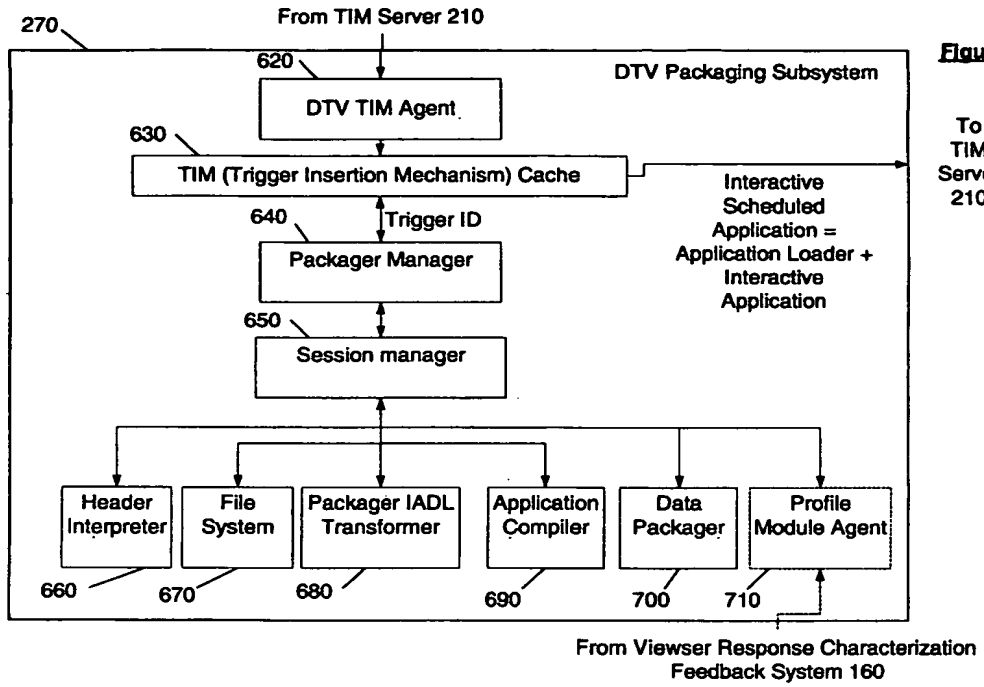


Figure 13

Figure 14

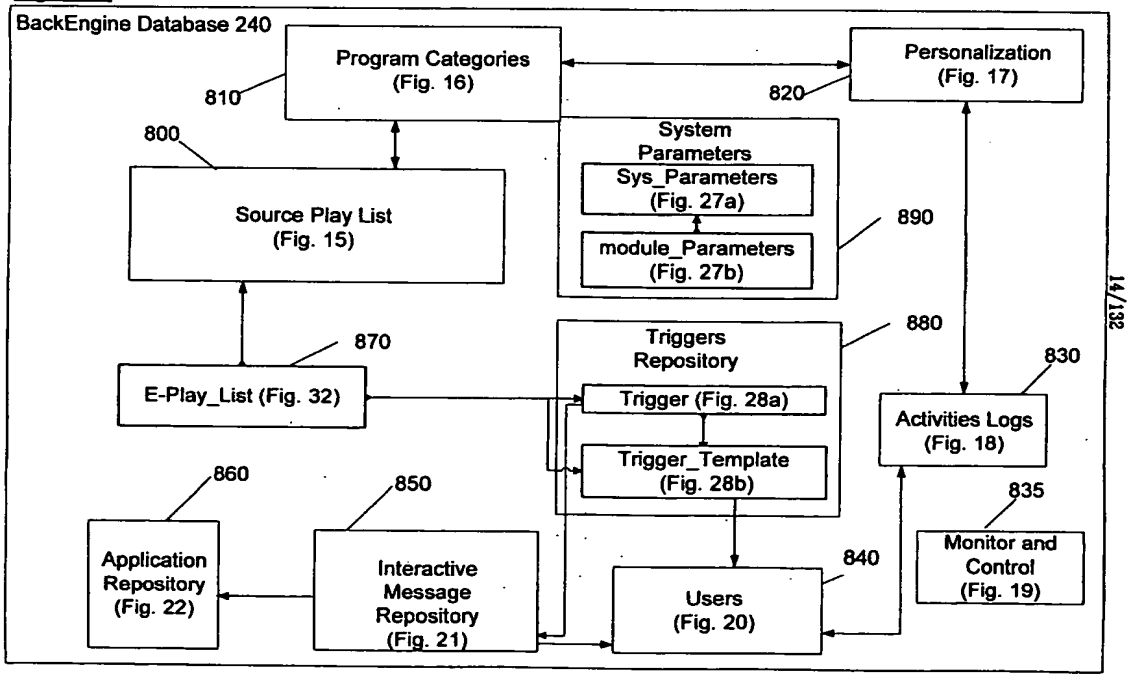


Figure 15

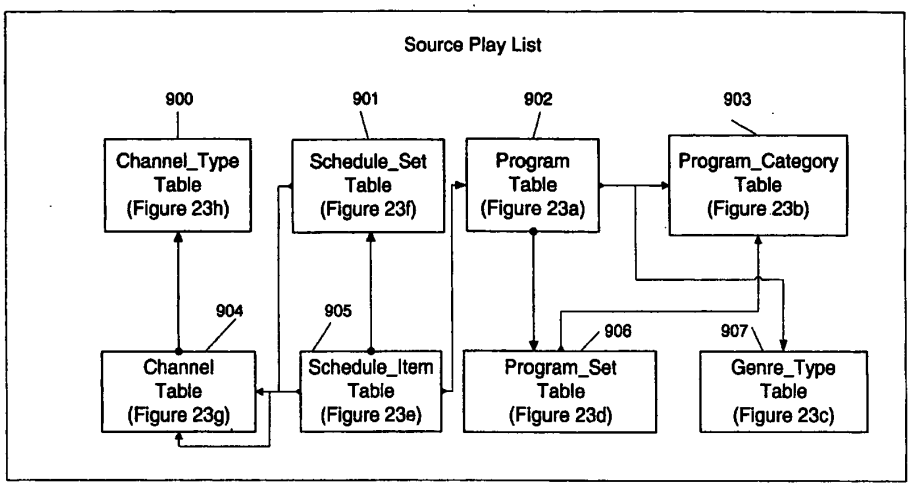


Figure 16

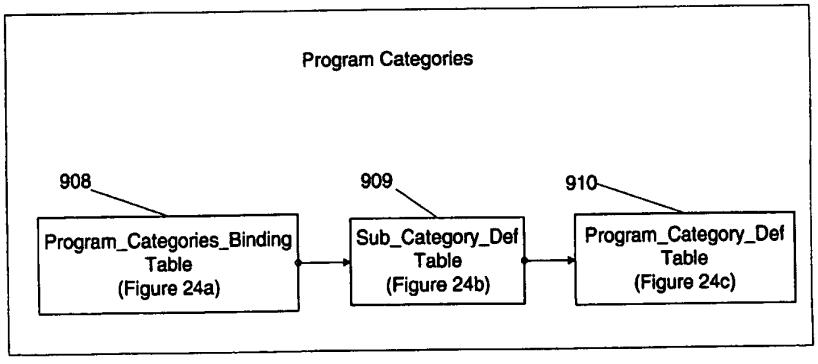


Figure 17

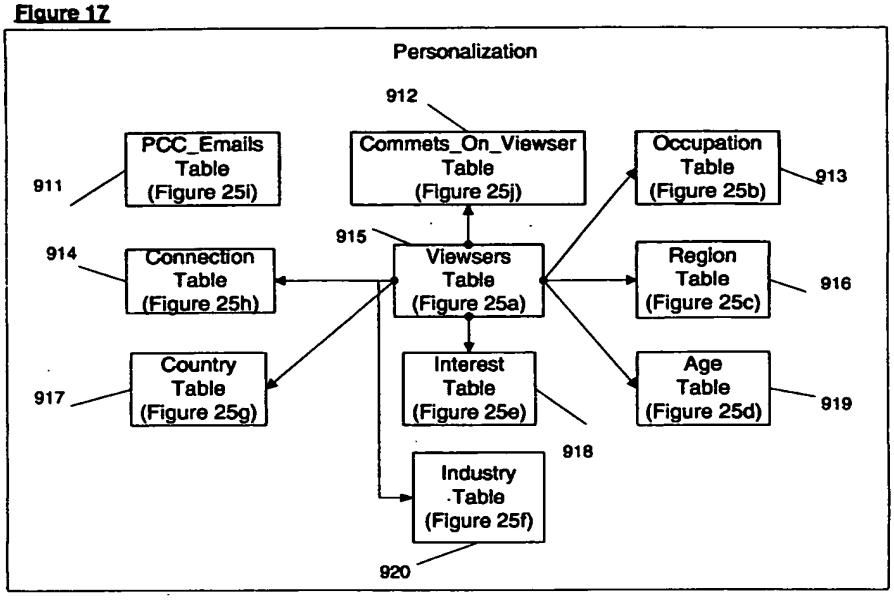


Figure 18

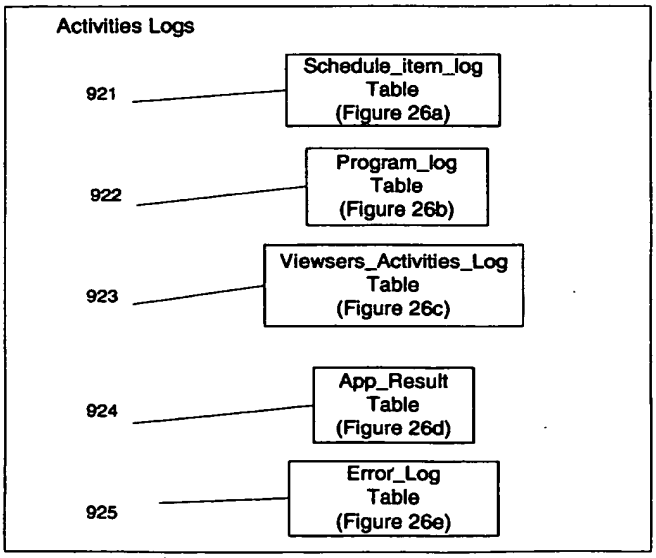


Figure 19

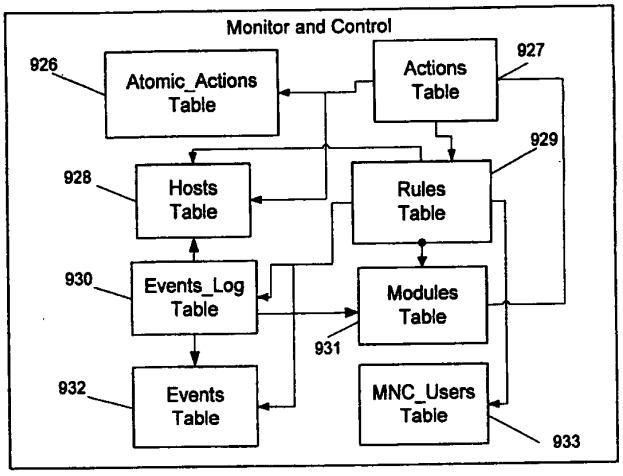


Figure 20

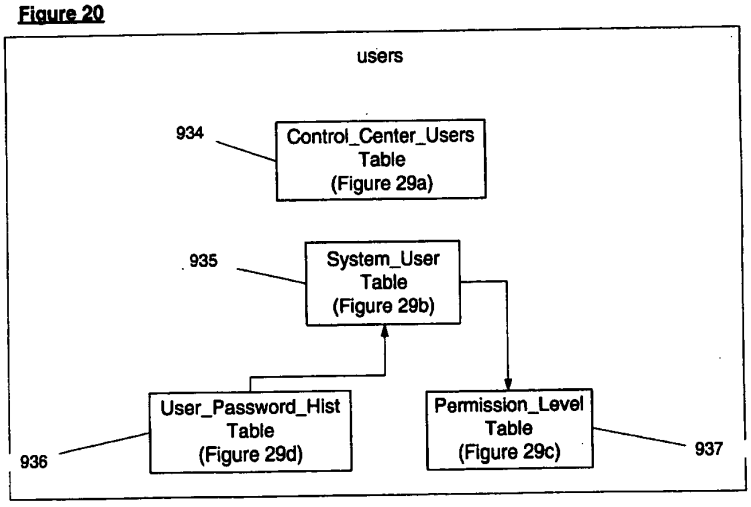


Figure 21

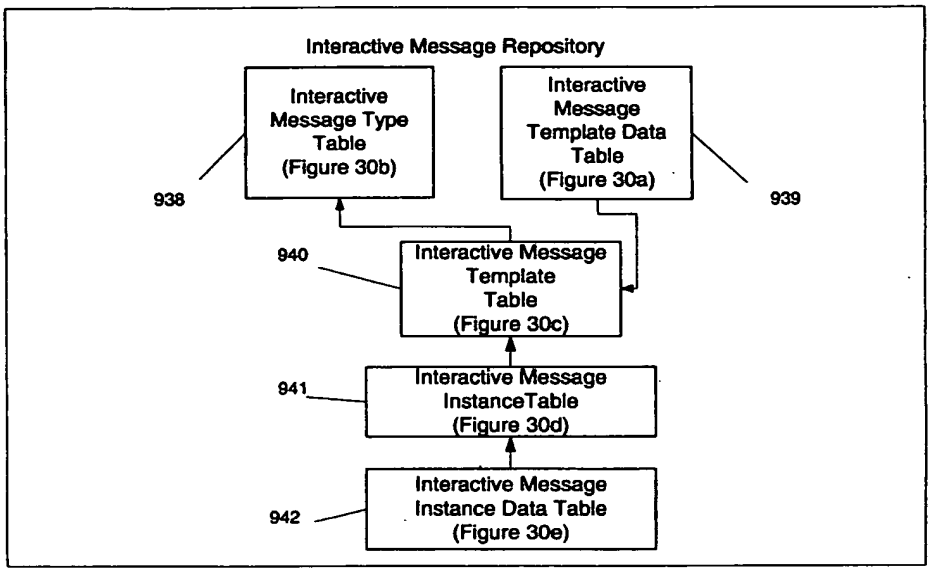


Figure 22

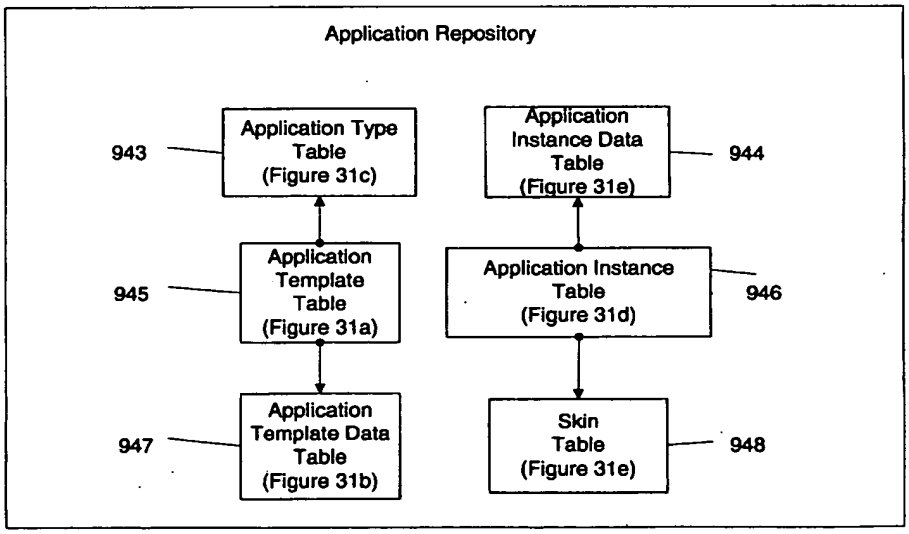


Figure 23a

Program Table (Fig. 15)

Column Name	Type	Primary Key	comment
PROGRAM_ID	VARCHAR2(64)	*	NOT NULL
PROG_SET_ID	NUMBER		
TITLE	VARCHAR2(255)		
ORIG_TITLE	VARCHAR2(255)		
GENRE1	NUMBER		
GENRE2	NUMBER		
PROD_YEAR	NUMBER		
COUNTRY	NUMBER		
DURATION	NUMBER		
EPISODE	NUMBER		
RERUN_FROM_DATE	DATE		
COLOR_FORMAT	NUMBER		
SOUND_FORMAT	NUMBER		
SUBTITLE_FOR_DEAF	VARCHAR2(255)		
CATEGORY	NUMBER		
PROD_COUNTRY	NUMBER		

Figure 23b

Program_Category Table (Fig. 15)

Column Name	Type	Primary Key	comment
CATEGORY_TYPE	NUMBER	*	NOT NULL
CATEGORY_DESC	VARCHAR2(255)		
TITLE	VARCHAR2(255)		

Figure 23c

Genre_Type Table (Fig. 15)

Column Name	Type	Primary Key	comment
GENRE_TYPE	NUMBER	*	NOT NULL
GENRE_NAME	VARCHAR2(255)		

Figure 23d

Program_Set Table (Fig. 15)

Column Name	Type	Primary Key	comment
PROG_SET_ID	NUMBER	*	NOT NULL
TITLE	VARCHAR2(255)		
ORIG_TITLE	VARCHAR2(255)		
GENRE1	NUMBER		
GENRE2	NUMBER		
PROD_YEAR	NUMBER		
COUNTRY	NUMBER		
CATEGORY	NUMBER		
NO_OF_EPISODES	NUMBER		

Figure 23e

Schedule_Item Table (Fig. 15)

Column Name	Type	Primary Key	comment
SCHEDULE_ID	VARCHAR2(64)	*	NOT NULL
SCH_SET_ID	NUMBER		
SCHEDULE_TIME	DATE		NOT NULL
DURATION	NUMBER		
PROGRAM_ID	VARCHAR2(64)		NOT NULL
PART_NO	NUMBER		
BROADCAST_TYPE	VARCHAR2(255)		
LIVE	VARCHAR2(1)		
PREVIEW_NAME	VARCHAR2(255)		
PREVIEW_START_TIME	NUMBER		
PREVIEW_END_TIME	NUMBER		
ORIG_ID	VARCHAR2(32)		
MATERIAL_ID	VARCHAR2(32)		
ON_HOLD	NUMBER(1)		
HIDDEN			NUMBER(1)
GAP	NUMBER(1)		
TIME_STAMP	DATE		
SCHEDULE_GROUP	NUMBER		

Figure 23f

SCHEDULE_SET Table (Fig. 15)

Column Name	Type	Primary Key	comment
SCH_SET_ID	NUMBER	*	NOT NULL
CHANNEL_ID	NUMBER		
SCHEDULE_DATE	DATE		NOT NULL
START_TIME	DATE		NOT NULL
DURATION	NUMBER		
SPECIAL_ATT	VARCHAR2(255)		
BROADCAST_TYPE	VARCHAR2(255)		
SENDER	VARCHAR2(255)		
ADDRESSEE	VARCHAR2(255)		
CREATION_DATE	DATE		
STATUS	VARCHAR2(255)		

Figure 23g

ChannelTable (Fig. 15)

Column Name	Type	Primary Key	comment
CHANNEL_ID	NUMBER	*	NOT NULL
NAME	VARCHAR2(255)		
STATION_ID	NUMBER		
CH_DESC	VARCHAR2(4000)		
CHANNEL_TYPE	NUMBER		

Figure 23h

ChannelType Table (Fig. 15)

Column Name	Type	Primary Key	comment
CHANNEL_TYPE	NUMBER	*	NOT NULL
TYPE_NAME	VARCHAR2(255)		

Figure 24a

Program_Categories_Binding Table (Fig. 16)

Column Name	Type	Primary Key	comment
PROGRAM_ID	VARCHAR2(64)	*	NOT NULL
CATEGORY_ID	NUMBER	*	NOT NULL
SUB_CAT_ID	NUMBER	*	NOT NULL
TIME_STAMP	DATE		
CREATED_BY	NUMBER		

Figure 24b

Sub_Category_Def Table (Fig. 16)

Column Name	Type	Primary Key	comment
SUB_CAT_ID	NUMBER	*	NOT NULL
CATEGORY_ID	NUMBER	*	NOT NULL
TITLE	VARCHAR2(255)		NOT NULL
DESCRIPTION	VARCHAR2(255)		

Figure 24c

Program_Category_Def Table (Fig. 16)

Column Name	Type	Primary Key	comment
CATEGORY_ID	NUMBER	*	NOT NULL
TITLE	VARCHAR2(255)		NOT NULL
DESCRIPTION	VARCHAR2(255)		
TIME_STAMP	DATE		
CREATED_BY	NUMBER		

Figure 25a

Viewsers Table (Fig. 17)

Column Name	Type	Primary Key	comment
USER_ID	NUMBER	*	NOT NULL
USER_NAME	VARCHAR2(12)		NOT NULL
PASSWORD	VARCHAR2(12)		NOT NULL
FIRST_NAME	VARCHAR2(32)		NOT NULL
MIDDLE_NAME	VARCHAR2(32)		
LAST_NAME	VARCHAR2(32)		NOT NULL
AGE_INDEX	NUMBER		
GENDER	VARCHAR2(1)		
ADDRESS	VARCHAR2(32)		
CITY	VARCHAR2(32)		
STATE	NUMBER		
ZIP_CODE	NUMBER(5)		
REGION	NUMBER		
COUNTRY	NUMBER		
PHONE_NUMBER	VARCHAR2(32)		
MOBILE	VARCHAR2(32)		
FAX	VARCHAR2(32)		
OCCUPATION	NUMBER		
EMAIL	VARCHAR2(255)		NOT NULL
IP_ADDRESS	VARCHAR2(15)		
PILOT_GROUP	NUMBER(1)		
INDUSTRY	NUMBER		
INTERESTS	VARCHAR2(255)		
REGISTRATION_DATE	DATE		NOT NULL
PASSWORD_CHANGED_DATE	DATE		NOT NULL
LAST_UPDATE_TIME	DATE		NOT NULL
REGISTERED_BY	NUMBER		
CONNECTION	NUMBER		

Figure 25b

Occupation Table (Fig. 17)

Column Name	Type	Primary Key	Comment
OCCUP_ID	NUMBER	*	NOT NULL
TITLE	VARCHAR2(255)		NULL

Figure 25c

Region Table (Fig. 17)

Column Name	Type	Primary Key	comment
REGION_ID	NUMBER	*	NOT NULL
REGION_NAME	VARCHAR2(255)		NOT NULL

Figure 25d

AgeTable (Fig. 17)

Column Name	Type	Primary Key	Comment
AGE_ID	NUMBER	*	NOT NULL
RANGE	VARCHAR2(12)		VARCHAR2(12)

Figure 25e

Interest Table (Fig. 17)

Column Name	Type	Primary Key	comment
INTEREST_ID	NUMBER	*	NOT NULL
TITLE	VARCHAR2(255)		NOT NULL

Figure 25f

Industry Table (Fig. 17)

Column Name	Type	Primary Key	comment
INDUSTRY_ID	NUMBER	*	NOT NULL
TITLE	VARCHAR2(255)		NOT NULL

Figure 25g

Country Table (Fig. 17)

Column Name	Type	Primary Key	comment
COUNTRY_ID	NUMBER	*	NOT NULL
COUNTRY_NAME	VARCHAR2(255)		

Figure 25h

Connection Table (Fig. 17)

Column Name	Type	Primary Key	comment
CONNECTION_ID	NUMBER	*	NOT NULL
CONNECTION_TYPE	VARCHAR2(32)		NOT NULL

Figure 25i

PCC_Emails Table (Fig. 17)

Column Name	Type	Primary Key	comment
MAIL_ID	NUMBER	*	NOT NULL
SENDER	NUMBER		
SENT_DATE	DATE		
SUBJECT	VARCHAR2(255)		
ATTACHMENT_NAME	VARCHAR2(32)		
MAIL_CONTENT	CLOB		
PILOT_GROUP	NUMBER(1)		

Figure 25j

Comments_On_Viewsers Table (Fig. 17)

Column Name	Type	Primary Key	comment
CHANNEL_TYPE	NUMBER		NOT NULL
TYPE_NAME	VARCHAR2(255)		
COMMENT_ID	NUMBER	*	NOT NULL
VIEWSER_ID	NUMBER		NOT NULL
SUBJECT	VARCHAR2(255)		NOT NULL
TEXT	VARCHAR2(2000)		
COMMENT_DATE	DATE		NOT NULL
COMMENT_BY	NUMBER		NOT NULL

Figure 26a

Schedule_Item_Log Table (Fig. 18)

Column Name	Type	Primary Key	Comment
LOG_ID	NUMBER	*	NOT NULL
CHANNEL_ID	NUMBER		NULL
SENDER	VARCHAR2(255)		NULL
ADDRESSEE	VARCHAR2(255)		NULL
SCH_START_TIME	DATE		NULL
SCH_END_TIME	DATE		NULL
START_TIME	DATE		NULL
END_TIME	DATE		NULL
UPDATE_BY	VARCHAR2(255)		NULL
STATUS	VARCHAR2(16)		NULL
VERSION	VARCHAR2(16)		NULL

Figure 26b

PROGRAM_LOG Table (Fig. 17)

Column Name	Type	Primary Key	Comment
LOG_ID	NUMBER	*	NOT NULL
CHANNEL_ID	NUMBER		NULL
SENDER	VARCHAR2(255)		NULL
ADDRESSEE	VARCHAR2(255)		NULL
START_TIME	DATE		NULL
END_TIME	DATE		NULL
UPDATE_BY	VARCHAR2(255)		NULL
STATUS	VARCHAR2(16)		NULL
VERSION	VARCHAR2(16)		NULL

Figure 26c

Viewsers_activites_Log Table (Fig. 17)

Column Name	Type	Primary Key	Comment
VIEWSER_ID	NUMBER	*	NOT NULL
ACTIVITY_ID	NUMBER	*	NOT NULL
IP_ADDRESS	VARCHAR2(15)		NULL
LOG_DATE	DATE	*	NOT NULL
APP_INST_ID	NUMBER		NULL
APP_STAGE_NUM	NUMBER		NULL
BROWSER	VARCHAR2(255)		NULL
FREE_TEXT	VARCHAR2(4000)		NULL

Figure 26d

App_Result Table (Fig. 17)

Column Name	Type	Primary Key	Comment
survey_id	NUMBER	*	NOT NULL
question_id	NUMBER	*	NOT NULL
answer1	NUMBER		NULL
answer2	NUMBER		NULL
answer3	NUMBER		NULL
answer4	NUMBER		NULL
answer5	NUMBER		NULL
answer6	NUMBER		NULL
answer7	NUMBER		NULL
answer8	NUMBER		NULL
answer9	NUMBER		NULL
answer10	NUMBER		NULL
first_vote_date	DATE		NULL
last_vote_date	DATE		NULL

Figure 26e

Error_Log Table (Fig. 17)

Column Name	Type	Primary Key	Comment
Err_ID	NUMBER	*	NOT NULL
Time_stamp	DATE		NOT NULL

Figure 27a

Sys_Parameters Table (Fig. 14)

Column Name	Type	Primary Key	comment
PARAM_ID	NUMBER		NOT NULL
PARAM_NAME	VARCHAR2(255)		NOT NULL
PARAM_VALUE	VARCHAR2(255)		NOT NULL
LAST_MODIFY_DATE	DATE		NOT NULL
DESCRIPTION	VARCHAR2(255)		

Figure 27b

Module_Parameters Table (Fig. 14)

Column Name	Type	Primary Key	Comment
Module_ID	NUMBER	*	NOT NULL
Param_ID	NUMBER	*	NOT NULL

Figure 28a

Triggers Table (Fig. 14)

Column Name	Type	Primary Key	comment
TRG_ID	NUMBER	*	NOT NULL
TITLE	VARCHAR2(255)		NOT NULL
IM_INST_ID	NUMBER		
STATUS	NUMBER		
CREATED_BY	NUMBER		
CREATE_DATE	DATE		
CHANGED_BY	NUMBER		
CHANGE_DATE	DATE		
ICON_PATH	VARCHAR2(255)		
REPOSITORY	NUMBER(1)		

Figure 28b

Trigger_Template Table (Fig. 14)

Column Name	Type	Primary Key	comment
TRG_TMPL_ID	NUMBER	*	NOT NULL
TITLE	VARCHAR2(255)		NOT NULL
PROG_ID	VARCHAR2(64)		
TRG_ID	NUMBER		
DAY_IN_MONTH	VARCHAR2(255)		
DAY_IN_WEEK	VARCHAR2(50)		
CREATED_BY	NUMBER		
CREATE_DATE	DATE		
CHANGED_BY	NUMBER		
CHANGE_DATE	DATE		
START_DATE	DATE		
END_DATE	DATE		
CONFIRMED	NUMBER		
OCCURRENCES	NUMBER		
START_RELATIVE_TIME	NUMBER		
DURATION	NUMBER		
INTERVAL	NUMBER		
ALREADY_OCCURRED	NUMBER		

Figure 29a

Control_Center_User Table (Fig. 20)

Column Name	Type	Primary Key	comment
USER_ID	NUMBER	*	NOT NULL
USER_NAME	VARCHAR2(12)		NOT NULL
PASSWORD	VARCHAR2(12)		NOT NULL
FIRST_NAME	VARCHAR2(32)		
LAST_NAME	VARCHAR2(32)		
POSITION	VARCHAR2(255)		
COMPANY	VARCHAR2(255)		
TELEPHONE	VARCHAR2(255)		
MOBILE	VARCHAR2(255)		
LAST_MODIFY_DATE	DATE		NOT NULL
PASSWORD_CHANGED_DATE	DATE		NOT NULL

Figure 29b

System_User Table (Fig. 20)

Column Name	Type	Primary Key	comment
USER_ID	NUMBER	*	NOT NULL
USER_NAME	VARCHAR2(30)		NOT NULL
PASSWORD	VARCHAR2(255)		NOT NULL
PASSWORD_CHANGED_DATE	DATE		
PERMISSION_LEVEL	NUMBER		NOT NULL
FIRST_NAME	VARCHAR2(255)		
LAST_NAME	VARCHAR2(255)		
ENABLE	NUMBER(1)		NOT NULL

Figure 29c

Permission_Level Table (Fig. 20)

Column Name	Type	Primary Key	comment
PERMISSION_LEV	NUMBER	*	NOT NULL
PERMISSION_NAME	VARCHAR2(255)		NOT NULL
PERMISSION_DESC	VARCHAR2(255)		

Figure 29d

USER_PASSWORD_HIST Table (Fig. 20)

Column Name	Type	Primary Key	Comment
USER_ID	NUMBER	*	NOT NULL
PASSWORD_DATE	DATE		NOT NULL
PASSWORD_VALUE	NUMBER		NOT NULL

Figure 30a

IM Template data_Table (Fig. 21)

Column Name	Type	Primary Key	Comment
Title	varchar2(128)	*	
platform	number		
file_Type	number		
IMG	ordsys.ordimage		
AUD	ordsys.ordaudio		
VID	ordsys.ordvideo		
char_file	clob		
binary_file	blob		
Time_stamp	date		

Figure 30b

Interactive_Message_Type Table (Fig. 21)

Column Name	Type	Primary Key	comment
IM_TYPE	NUMBER	*	NOT NULL
IM_TYPE_NAME	VARCHAR2(255)		

Figure 30c

Interactive_Message_Template Table (Fig. 21)

Column Name	Type	Primary Key	comment
IM_TMPL_ID	NUMBER	*	NOT NULL
TITLE	VARCHAR2(255)		
HEADER_XML	CLOB		
MAIN_XML	CLOB		
MEDIA	TMEDIA		
IM_TYPE	NUMBER		NOT NULL
CREATED_BY	NUMBER		
CREATE_DATE	DATE		
CHANGED_BY	NUMBER		
CHANGE_DATE	DATE		
ICON_PATH	VARCHAR2(255)		
PLATFORM	NUMBER		

Figure 30d

Interactive_Message_Instance Table (Fig. 21)

Column Name	Type	Primary Key	comment
IM_INST_ID	NUMBER	*	NOT NULL
TITLE	VARCHAR2(255)		
IM_ID	NUMBER		
APP_INST_ID	NUMBER		
IM_TEXT	VARCHAR2(4000)		
ICON_PATH	VARCHAR2(255)		
CREATED_BY	NUMBER		
CREATE_DATE	DATE		
CHANGED_BY	NUMBER		
CHANGE_DATE	DATE		
REPOSITORY	NUMBER(1)		
HEADER_XML	CLOB		
DATA_XML	CLOB		
MEDIA	TMEDIA		

Figure 30e

IM_Instance_Data Table (Fig. 21)

Column Name	Type	Primary Key	Comment
Title	varchar2(128(		
platform	number		
file_Type	number		
IMG	ordsys.ordimage		
AUD	ordsys.ordaudio		
VID	ordsys.ordvideo		
char_file	clob		
binary_file	blob		
Time_stamp	date		

Figure 31a

Application_Template Table (Fig. 22)

Column Name	Type	Primary Key	Comment
APP_TMPL_ID	NUMBER	*	NOT NULL
TITLE	VARCHAR2(255)		
APP_TYPE_ID	NUMBER		NOT NULL
XML	CLOB		
TMPL_DESC	VARCHAR2(255)		
ICON_PATH	VARCHAR2(255)		
CREATED_BY	NUMBER		
CREATE_DATE	DATE		
CHANGED_BY	NUMBER		
CHANGE_DATE	DATE		
HEADER_XML	CLOB		
MAIN_XML	CLOB		
MEDIA	TMEDIA		
PLATFORM	NUMBER		

Figure 31b

Application_Template_Data Table (Fig. 22)

Column Name	Type	Primary Key	Comment
Title	varchar2(128)	*	
platform	number		
file_Type	number		
IMG	ordsys.ordimage		
AUD	ordsys.ordaudio		
VID	ordsys.ordvideo		
char_file	clob		
binary_file	blob		
Time_stamp	date		

Figure 31c

Application_Type Table (Fig. 22)

Column Name	Type	Primary Key	comment
APP_TYPE_ID	NUMBER	*	NOT NULL
TITLE	VARCHAR2(255)		NOT NULL
APP_TYPE_DESC	VARCHAR2(255)		
XML	CLOB		
CREATED_BY	NUMBER		
CREATE_DATE	DATE		
CHANGED_BY	NUMBER		
CHANGE_DATE	DATE		
EVENT	NUMBER		NOT NULL

Figure 31d

Application_Instance Table (Fig. 22)

Column Name	Type	Primary Key	Comment
APP_INST_ID	NUMBER	*	NOT NULL
TITLE	VARCHAR2(255)		
APP_DATA_ID	NUMBER		
SKIN_ID	NUMBER		
ICON_PATH	VARCHAR2(255)		
CREATED_BY	NUMBER		
CREATE_DATE	DATE		
CHANGED_BY	NUMBER		
CHANGE_DATE	DATE		
REPOSITORY	NUMBER(1)		
STATUS	NUMBER(1)		DEFAULT0
XML	CLOB		
APP_TMPL_ID	NUMBER		NOT NULL
RC	CLOB		
HEADER_XML	CLOB		
DATA_XML	CLOB		
MEDIA	TMEDIA		

Figure 31e

Application_instance_data Table (Fig. 22)

Column Name	Type	Primary Key	Comment
Title	varchar2(128)	*	
platform	number		
file_Type	number		
IMG	ordsys.ordimage		
AUD	ordsys.ordaudio		
VID	ordsys.ordvideo		
char_file	clob		
binary_file	blob		
Time_stamp	date		

Figure 31f

Skin Table (Fig. 22)

Column Name	Type	Primary Key	comment
SKIN_ID	NUMBER	*	NOT NULL
TITLE	VARCHAR2(255)		
SKIN_PATH	VARCHAR2(2000)		
SKIN_XML	CLOB		
ICON_PATH	VARCHAR2(255)		
CREATED_BY	NUMBER		
CREATE_DATE	DATE		
CHANGED_BY	NUMBER		
CHANGE_DATE	DATE		
TIME_STAMP	DATE		
CREATED_BY	NUMBER		

Figure 32

E_Play_List Table (Fig. 14)

Column Name	Type	Primary Key	comment
SCHEDULE_ID	VARCHAR2(64)		
TRG_ID	NUMBER		NOT NULL
TRG_START_RELATIVE_TIME	NUMBER		
TRG_END_RELATIVE_TIME	NUMBER		
WEB_JOCKEY_IND	NUMBER		
CREATED_BY	NUMBER		
CREATE_DATE	DATE		
CHANGED_BY	NUMBER		
CHANGE_DATE	DATE		
CONFIRMED	NUMBER(1)		NOT NULL
START_TIME	DATE		
DURATION	NUMBER		
TRG_TMPL_ID	NUMBER		

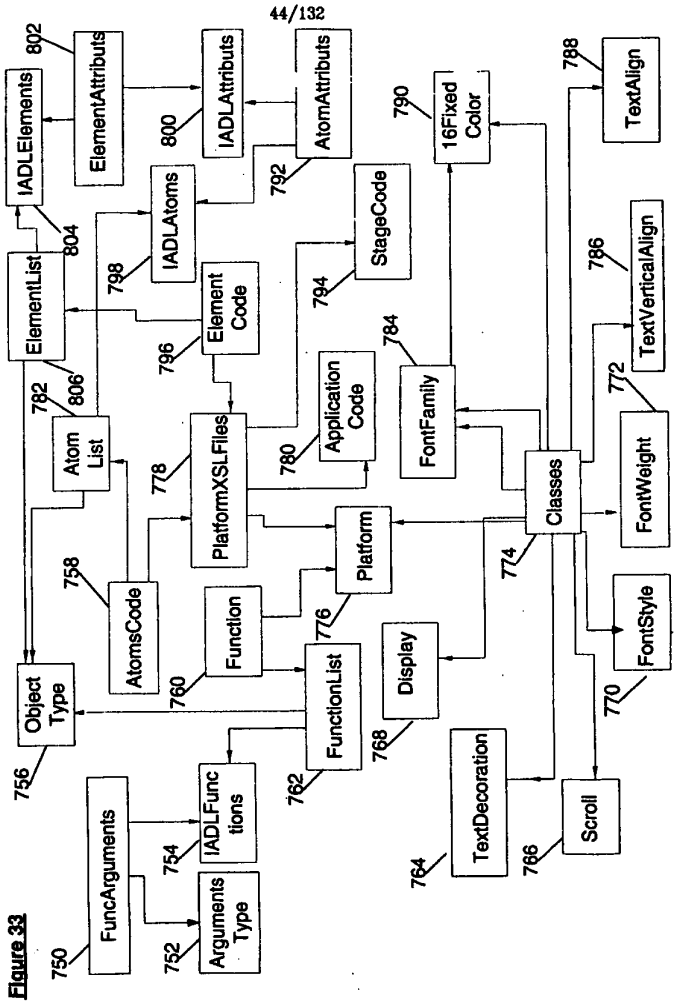


Figure 33

Figure 34A

Procedure #	Procedure Name	Input Parameters	Output Parameters	Description
11001	NewTrgInstance	TRG_ID (Number), IM_Tmpl_ID (Number), User_ID (Number)	New_Trg_ID (Number)	Creates New Action. Input To Tables: Triggers, Interactive_Message_Instance, Application_Instance.
11002	SetTrgStatus	Trg_ID (Number), App_Or_IM (Number), User_ID (Number)	O_Notify (Number) = [1 – O.K, (-1) – Err]	Set The Action status 0 for update (delete instance files). Tables : Triggers (status), App_Inst_Data
12001	GetAppPopulation Data	TRG_ID (Number), Is_New(Number)	Record Sets Of: File_Name (Varchar2), File_Data (CLOB)	Gets Application Files : Header.xml Main.xml, Inst_Header.xml(empty for template), Data.xml (empty for template), From Tables: Triggers, App_Tmpl_Data, App_Inst_Data

Figure 34B

Procedure #	Procedure Name	Input Parameters	Output Parameters	Description
12002	GetAppInstData	TRG_ID (Number), Is_New(Number), File_List (Varchar2)	Record Sets Of: File_Name (Varchar2), File_Data (BLOB) File_Data (CLOB)	Gets Application Instance Files. From Tables: Triggers, App_Tmpl_Data, App_Inst_Data
12003	SaveAppInstance File	TRG_ID (Number), File_Name (Varchar2),Character_Data (CLOB), Binary_Data(BLOB) File_Type (number)	O_Notify (Number) = [1 – O.K, (-1) – Err]	Loop For Insert Application Instance File Input To Tables : App_Inst_Data.

Figure 35A

Procedure #	Procedure Name	Input Parameters	Output Parameters	Description
20001	New_Trigger	i_user_ID , i_Title i_IM_Inst_ID i_APP_INST_ID i_Icon_Name i_Status	O_Trg_ID	Creates New Action. Input To Tables: Triggers, Interactive_Message_Instance, Application_Instance
20002	Duplicate_Trigger	i_user_ i_Source_Trg_ID	O_Trg_ID O_App_Inst_ID O_IM_Inst_ID	Duplicates Action. Input To Tables: Triggers, Interactive_Message_Instance, Application_Instance
20003	Update_Trigger	i_user_ID , I_Trg_ID, i_Title , i_IM_Inst_ID i_APP_INST_ID i_Icon_Name	O_NOTIFY [1 – O.K, (-1) – Err]	Update Action Information Input To Tables: Triggers.
20004	Set_Trigger_Repository	Trg_ID (Number), Repsitory (Varchar2), User_ID (Number)		Add Action to the Repository Input To Tables: Triggers.
20005	Set_Trigger_Title	Trg_ID (Number), Trg_Title (Varchar2), User_ID (Number)	O_Notify (Number) = [1 – O.K, (-1) – Err]	Set The Action Title Input To Tables: Triggers

Figure 35B

Procedure #	Procedure Name	Input Parameters	Output Parameters	Description
20006	Set_Trigger_Status	Trg_ID (Number), Status (Number), User_ID (Number)	O_Notify (Number) = [1 – O.K, (-1) – Err]	Set The Action status when completed Input To Tables: Triggers, Application_Instance
20007	Delete_Trigger	Trg_ID (Number), User_ID (Number)	O_Notify (Number) = [1 – O.K, (-1) – Err]	Delete Action. Input To Tables: Triggers, Interactive_Message_Instance, Application_Instance
60001	New_Recurrent_Trigger	i_User_Id i_Schedule_Id i_Trg_Id i_prog_id i_Start_Time i_end_Time i_Occurrences i_interval i_start_REL_time i_Duration i_webjokey i_Confirmed i_day i_month i_title	o_notify	Declaration of Recurrent Action Input To Tables: E_Play_List, Trigger_Template, Triggers, Interactive_Message_Instance, Application_Instance

WO 02/069121 PCT/IL02/00144

Figure 36

Procedure #	Procedure Name	Input Parameters	Output Parameters	Description
30001	Get_Triggers	user_ID Status	Triggers	Actions and Templates are received from triggers table

Figure 37

Procedure #	Procedure Name	Input Parameters	Output Parameters	Description
40001	Get_Play_List	i_start_date (DATE), i_end_date (DATE), i_user_ID (Number)	O_PlayList	Gets all Schedule Items (programs) Scheduled to the time range Output from tables: Schedule_Item, Program
40002	Get_Trigger_Details	i_start_date (DATE), i_end_date (DATE), i_user_ID (Number)	O_Triggers	Gets all Instances (triggers) assigned to the time or schedule items within the time range. Output from tables: Schedule_Item, E_Play_List, Triggers.
50001	New_EPL_Trigger	i_User_ID i_Trg_Id i_Schedule_Id i_Start_Time i_start_REL_time i_Duration i_Confirmed i_Event_ID	o_New_trg_id o_notify	Assign new instance (action) to time or shedule_item. Input To Tables: E_Play_List, Triggers, Interactive_Message_Instance, Application_Instance

Figure 38

Procedure #	Procedure Name	Input Parameters	Output Parameters	Description
70001	GetTriggersByTime	StartTime (Varchar2) EndTime (Varchar2)	Record Set Of: Sch_Id (Varchar2), Trg_Id (Number), Rel_Time (Number), Duration (Number), Abs_Time (Varchar2), trg_type (Number).	Gets both absolute and relative triggers either by time rang or by system parameter. Output from Tables: Schedule_Item, E_Play_List, Trigger_Template, Triggers
70002	GetTriggersByProgram	ScheduleID (Varchar2)	Record Set Of: Sch_Id (Varchar2), Trg_Id (Number), Rel_Time (Number), Duration (Number), Abs_Time (Varchar2)-NULL trg_type (Number).	Gets Relative triggers by ScheduleID (On Air) Output from Tables: Schedule_Item, E_Play_List, Trigger_Template, Triggers

Figure 39

Procedure #	Procedure Name	Input Parameters	Output Parameters	Description
80001	GetUniqueID		Unique_ID Number	Generate a unique ID for Applications and loaders compilations.
81001	GetHeaders	TRG_ID (Number), App_Type: (Number)	Record Set Of: Template_Header .xml + File_Name Instance_Header. xml xml + File_Name	Gets the Application template and instance IADL Headers Output from Tables: Interactive_Message_Template, Interactive_Message_Instance, Application_Template, Application_Instance
82001	GetTmplFiles	TRG_ID (Number),File_List (Varchar2),App_Type: (Number) NotifierInst AppInst	Record Sets Of: File_Name (Varchar2) File_Data (BLOB) File_Data(CLOB)	Gets the Application/Notifier Template Files. Output from Tables: IM_Tmpl_Data, App_Tmpl_Data.
82002	GetInstFiles	TRG_ID(Number), File_List (Varchar2),App_Type: (Number) NotifierInstAppInst	Record Sets Of: File_Name (Varchar2) File_Data(BLOB) File_Data(CLOB)	Gets the Application/Notifier Instance Files. Output from Tables: IM_Inst_Data, App_Inst_Data.

Figure 40

Procedure #	Procedure Name	Input Parameters	Output Parameters	Description
90001	VIEWSER_STAGES_LOG	(i_Viewser_Id, i_IP_Address, i_App_Inst_ID, i_App_Stage_Num, i_Browser, i_Free_Text		Write the viewser activities Input into Tables: VIEWSERS_ACTIVITIES_LOG

Figure 41

Procedure #	Procedure Name	Input Parameters	Output Parameters	Description
100001	New_ App_Template	i_App_Template_ Name, i_App_Type, i_Description	New Template ID	Create a new application template Input to Table: App_Tmpl_Data
100002	Save_ App_Tmpl_File	i_App_Tmpl_ID, i_File_Name, i_ Binary_Data, i_Char_Data, i_File_Type		Insert application template files. Input to Table: App_Tmpl_Data
100003	New_ IM_Template	i_IM_Template_N ame, i_IM_Type, i_Description	New Template ID	Create a new Notifier template Input to Table: Interactive_Message_Template, IM_Tmpl_Data
100004	Save_ IM_Tmpl_File	i_IM_Tmpl_ID, i_File_Name, i_Binary_Data, i_Char_Data, I_File_Type		Input to Table: IM_Tmpl_Data

Figure 42

Procedure #	Procedure Name	Input Parameters	Output Parameters	Description
110001	Viewser_Log	i_Viewser_Id, i_IP_Address, i_App_Inst_ID, i_App_Stage_Num, i_Browser, i_Free_Text		Input to Table: VIEWSERS_ACTIVITIES_LOG
110002	VIEWSER_Logins_LOG	i_Viewser_Id, i_IP_Address,		Verified Viewser Login. Input to Table: VIEWSERS_ACTIVITIES_LOG
110003	Viewser_Answer	i_Survey_ID, i_Question_ID, i_Answer_ID, O_Resaults		Write to Log the viewser answer. Input to Table: SURVEY_RESULTS

Figure 43

Procedure #	Procedure Name	Input Parameters	Output Parameters	Description
120001	Get_TIM_CACHE_BY_SCH_ID	i_Schedule_ID	Schedule_ID, trg_id, TRG_START_RELATIVE_TIME, Duration, IM_TEXT, APP_INST_ID, Event, SKIN_ID	Gets Relative triggers by ScheduleID (On Air) Output from Tables: Application_Type, Skins, Interactive_Message_Instance, E_play_list, Schedule_item, Triggers
120002	Get_TIM_CACHE_BY_Time	i_Start_Time, i_End_Time	Schedule_ID, Trg_id, TRG_START_RELATIVE_TIME, Duration, IM_TEXT, APP_INST_ID, Event, SKIN_ID	Gets both absolute and relative triggers either by time rang or by system parameter. Output from Tables: Application_Type, Skins, Interactive_Message_Instance, E_play_list, Schedule_item, Triggers.

Figure 44

Procedure #	Procedure Name	Input Parameters	Output Parameters	Description
130001	GET_APP_SKIN	i_user_ i_Skin_ID	O_AppSkinCur O_XML OUT	Gets The application skin Output from Tables: Skins
130002	GET_APP_INST_Data	i_user_ID i_App_Inst_ID	O_XML_cur AppInsData_Cur	Gets The application Output from Tables: Application_Instance

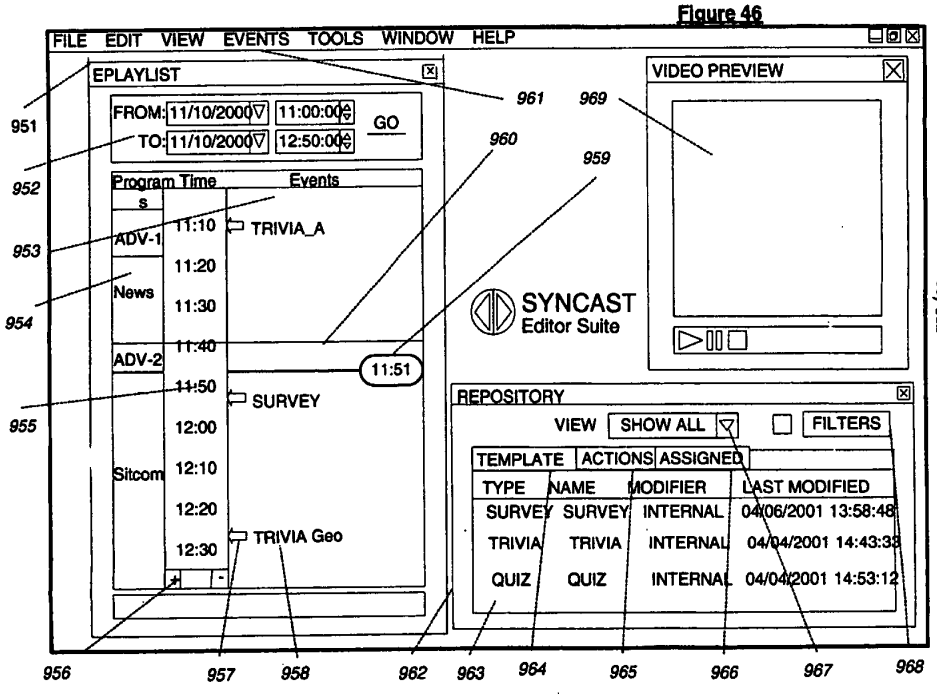
Figure 45A

Procedure #	Procedure Name	Input Parameters	Output Parameters	Description
140001	start_insert_program	i_channel_id, i_sender IN, i_addressee, i_update_by,i_STATUS, i_VERSION,	log_id	Input to Table: Program_Log
140002	Insert_Program	i_Program_id,i_Title, i_Orig_Title, i_Genre1, i_Genre2, i_Prod_Year, i_Duration,i_Episode_No, i_Rerun_From_Date, i_Color_Format, i_Sound_Format, i_Subtitle_For_Deaf, i_Category_Type, i_Country		Inputs are received from original third party Playlist management system (e.g. Pilat Media, Sepia) Input to Table: Program
140003	End_insert_program	i_log_id		Input to Table: Program_Log

Figure 45B

Procedure #	Procedure Name	Input Parameters	Output Parameters	Description
140004	start_insert_Sch_Item	i_channel_id, i_sender, i_addressee, i_sch_start_time, i_sch_end_time, i_update_by, i_STATUS, i_VERSION	O_log_id	Inputs are received from original third party Playlist management system (e.g. Pilat Media, Sepla) Input to Table: Sch_Item_Log
140005	Insert_Schedule_Item	i_Schedule_id i_Channel_id, i_Schedule_time, i_Duration, i_Program_id,BROADCAST_TYPE,i_Live,i_PREVIEW_NAME,i_PREVIEW_START_TIME,i_PREVIEW_END_TIME, i_ORIG_ID , i_MATERIAL_ID ,i_on_hold		Inputs are received from original third party Playlist management system (e.g. Pilat Media, Sepla) Input to Table: Schedule_Item
140006	End_insert_Sch_Item	i_log_id		Input to Table: Sch_Item_Log

Figure 46



WO 02/069121 PCT/IL02/00144

Figure 47

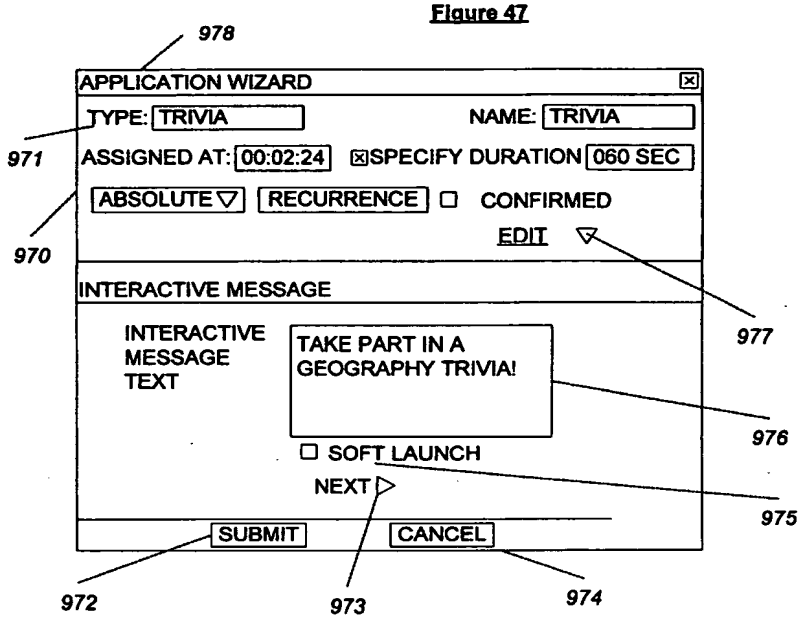


Figure 48

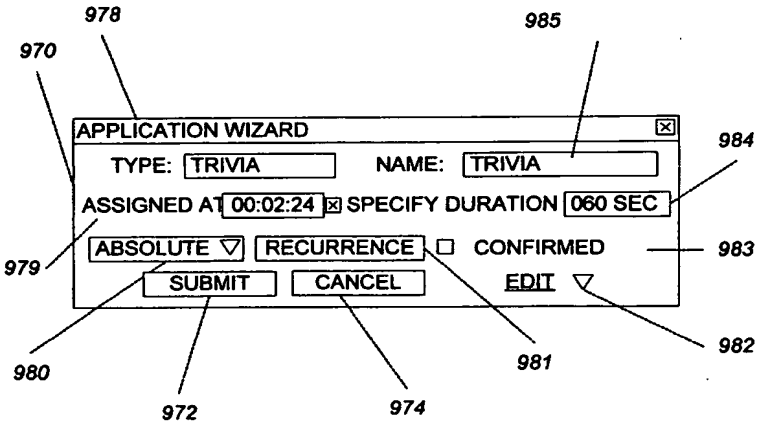


Figure 49

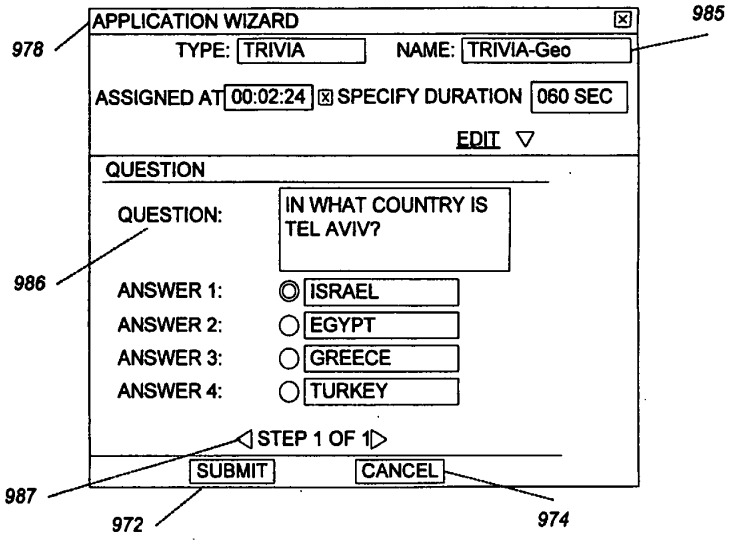


Figure 50

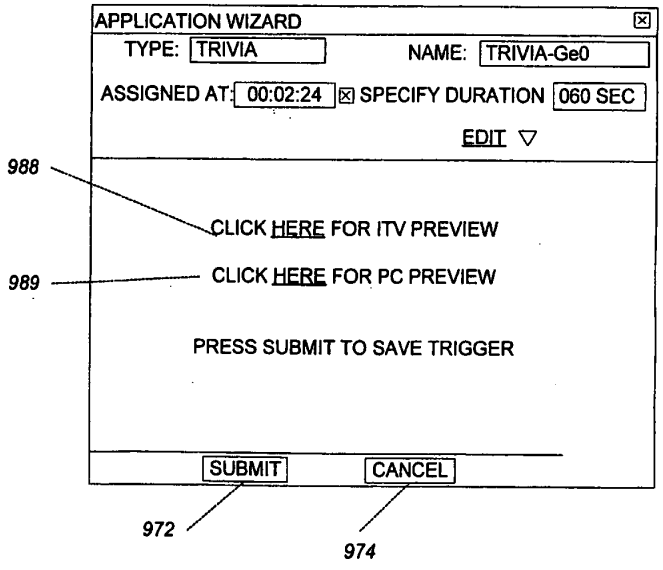


Figure 51

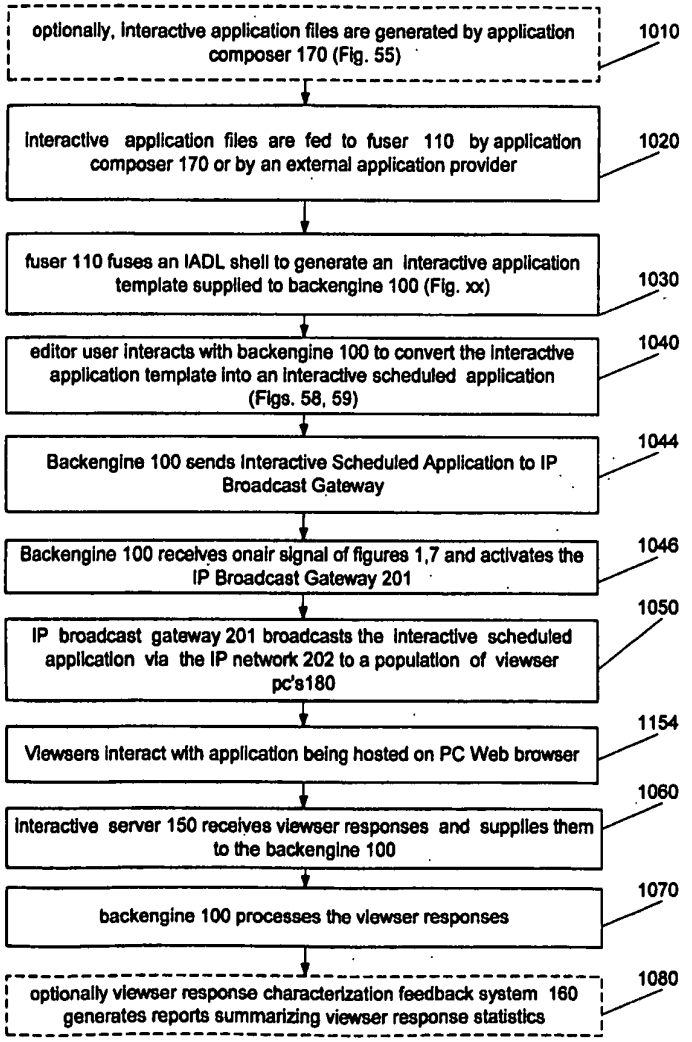


Figure 52

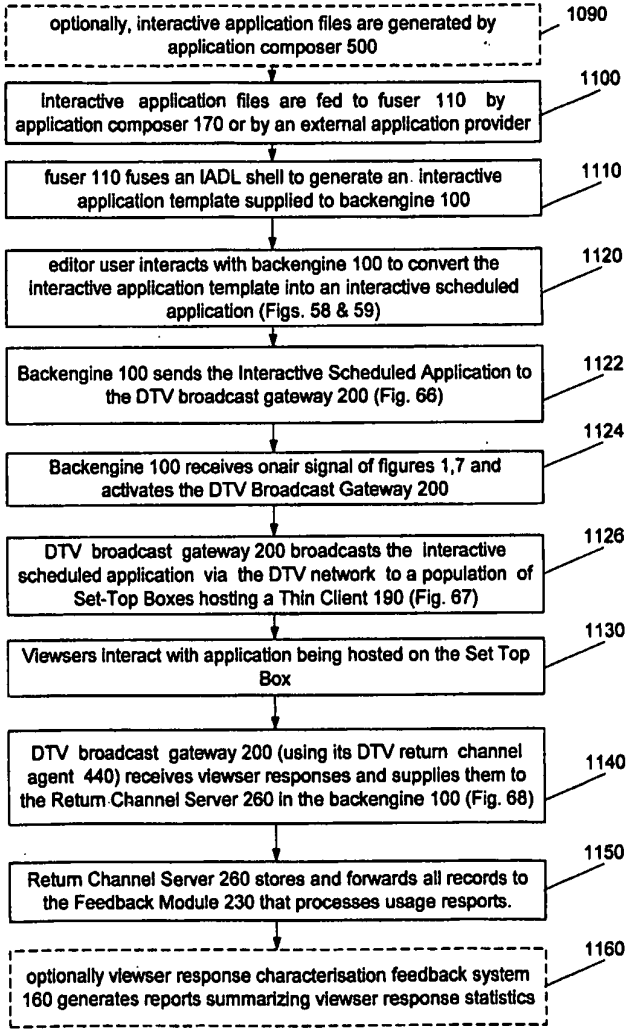


Figure 53A

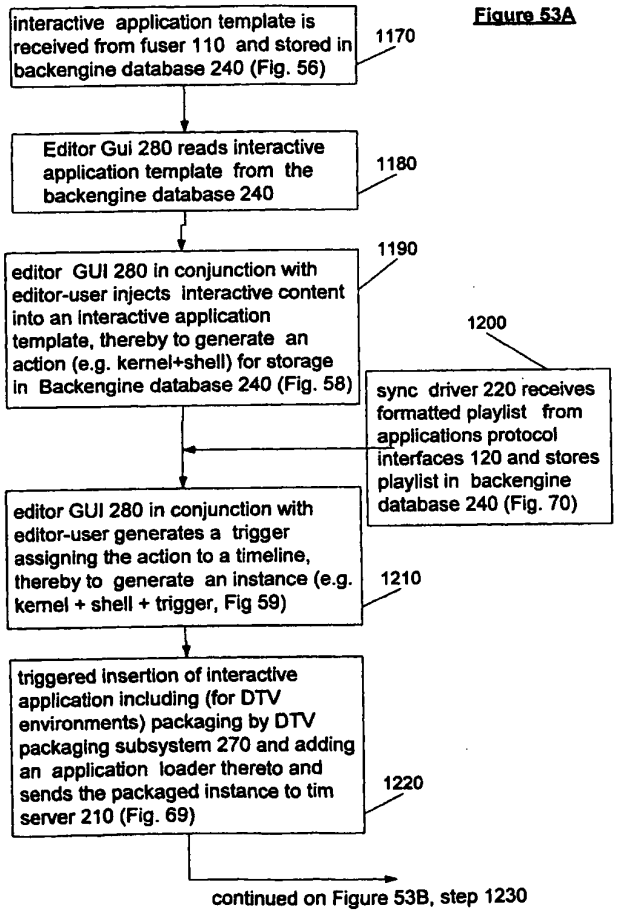


Figure 53B

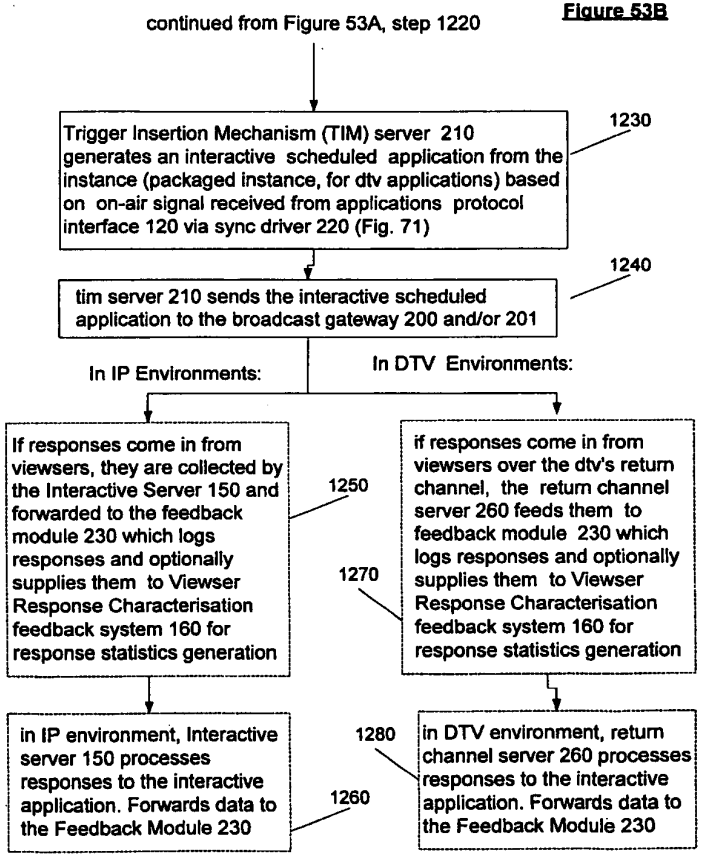


Figure 54

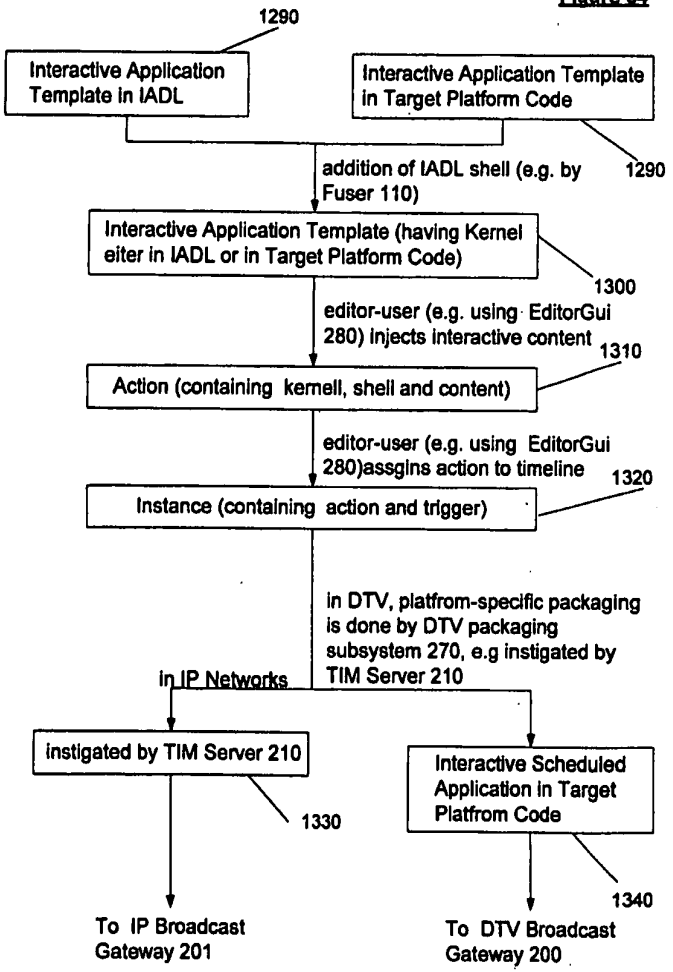


Figure 55

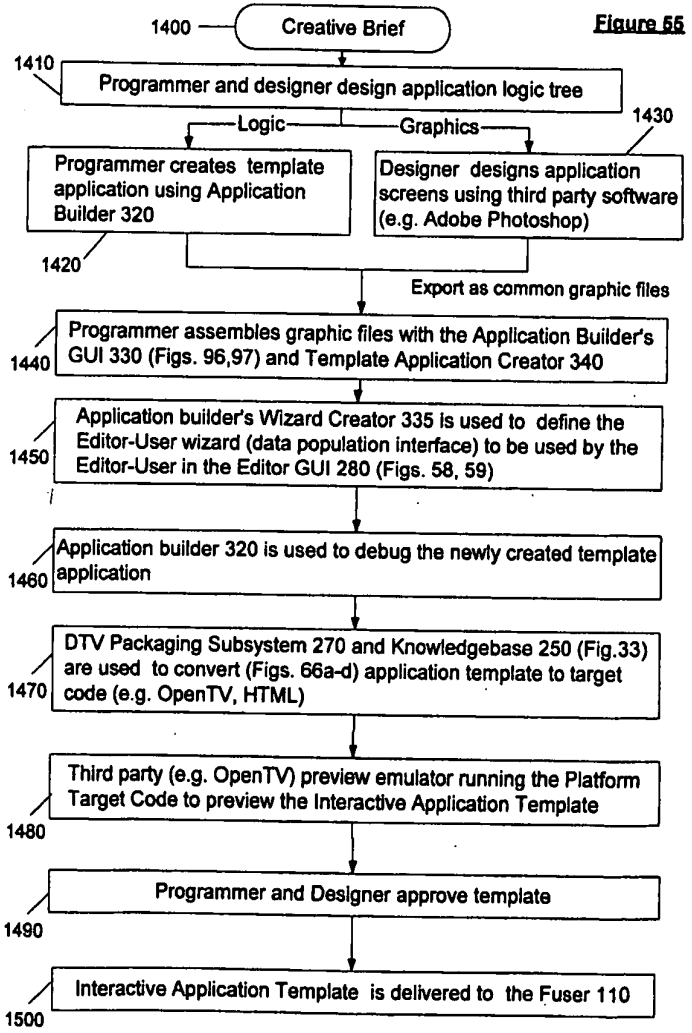


Figure 56

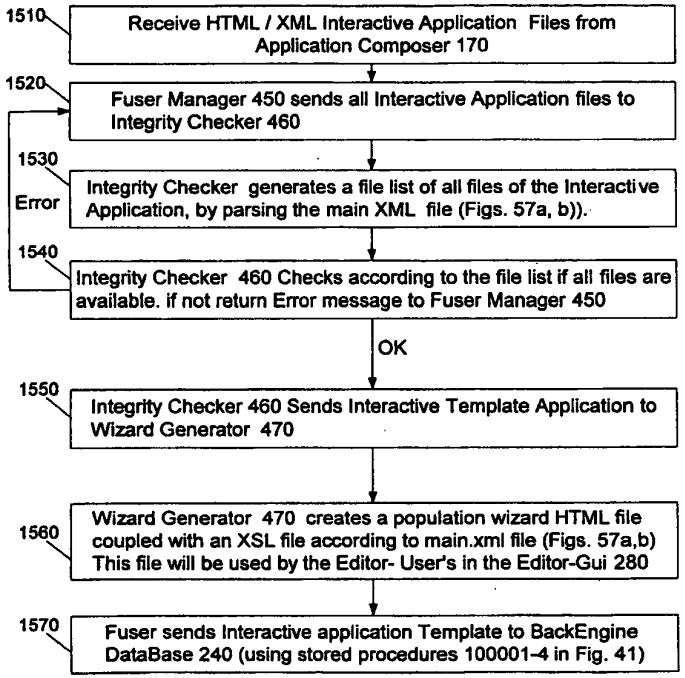


Figure 57A

```
<applicationDoc>
  <header>
    <properties>
      <name>More Info</name>
      <version>1.0</version>
      <description>More Info Application, Vertical Compact Mode.</description>
    </properties>
    <accessories>
      <screenMode>1</screenMode>
      <transMode>2</transMode>
      <transPercent>40</transPercent>
    </accessories>
  </header>
  <stage useGroupOfGroups="no">
    <shape groupOfGroups="no">
      <shapeBody normalClass="aboveLogoShape" shapeType="0"></shapeBody>
    </shape>
    <shape groupOfGroups="no">
      <shapeBody normalClass="rightToTitleShape" shapeType="0"></shapeBody>
    </shape>
    <image groupOfGroups="no">
      <imageSrc normalClass="pInfoTitleImage">reshetLogo.pix</imageSrc>
    </image>
    <textButton normalClass="menuTitleElem" groupOfGroups="no">
      <text normalClass="menuTitleAtom">
        <editInstructions title="Info title:" transformationType="textArea" input="true"
maxLength="8"/>
```

Figure 57B (cont. from 57A)

```
                </text>
            </textButton>
            <shape groupOfGroups="no">
                <shapeBody normalClass="rightToTextShape" shapeType="0"></shapeBody>
            </shape>
            <textArea normalClass="Elem" groupOfGroups="no" language="HEB" isFocus="yes"
direction="Vertical">
                <text normalClass="infoTextAtom" isFocus="yes">
                    <editInstructions title="Info:" transformationType="textArea" input="true" maxLength="50"/>
                </text>
                <imageSrc normalClass="ImageNavigationDummy" focusSrc=""></imageSrc>
                <imageSrc normalClass="ImageNavigationDummy" focusSrc=""></imageSrc>
            </textArea>
            <imageButton isFocus="no" groupOfGroups="no">
                <imageSrc isFocus="no" normalClass="quitApplication" keyType="keyRed">vertiExit.pix
                    <onClick>
                        <function>
                            <functionName>QuitApplication</functionName>
                        </function>
                    </onClick>
                </imageSrc>
            </imageButton>
            <shape groupOfGroups="no">
                <shapeBody normalClass="underQuitShape" shapeType="0"></shapeBody>
            </shape>
        </stage>
</applicationDoc>
```


Figure 58

Figure 59

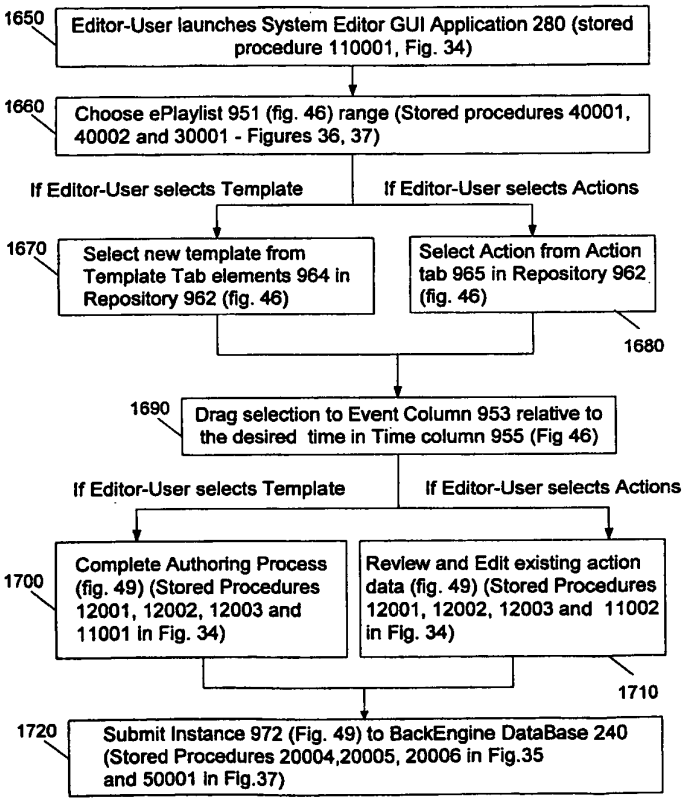


Figure 60

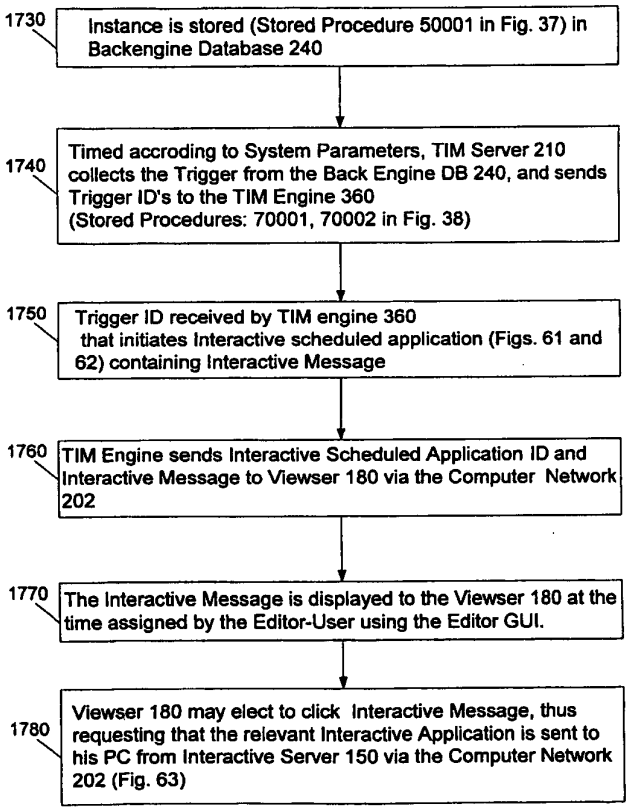


Figure 61

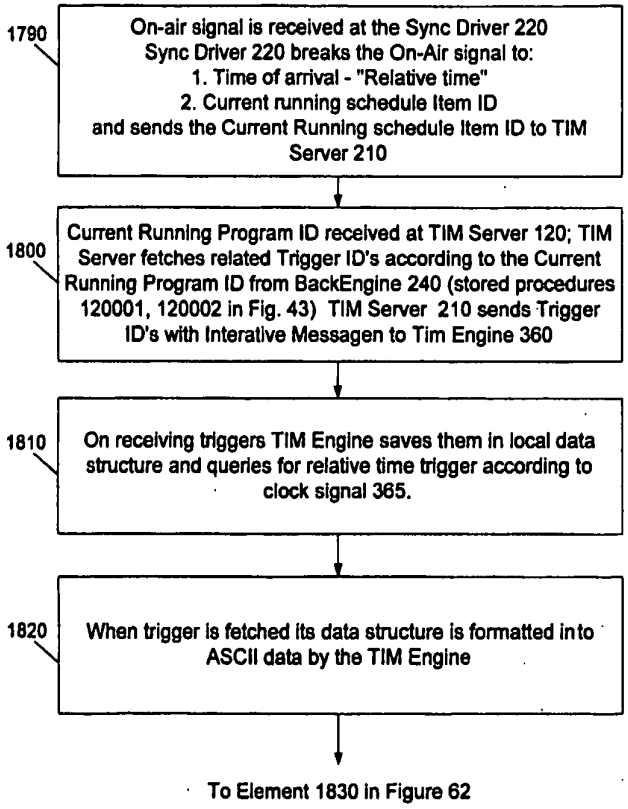


Figure 62

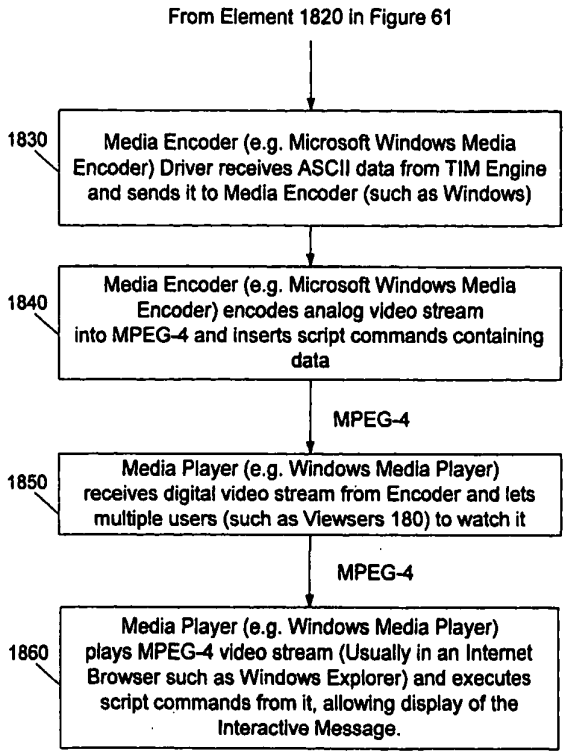


Figure 63

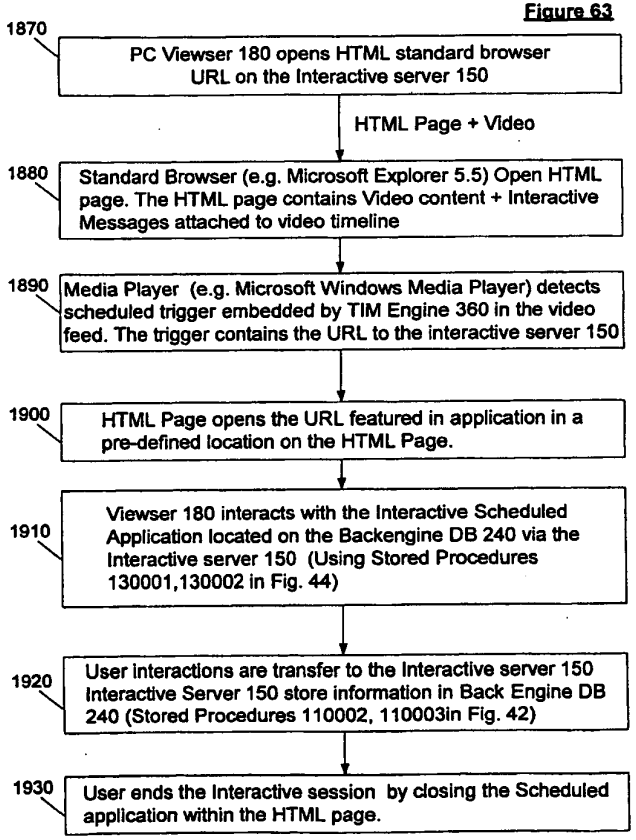


Figure 64

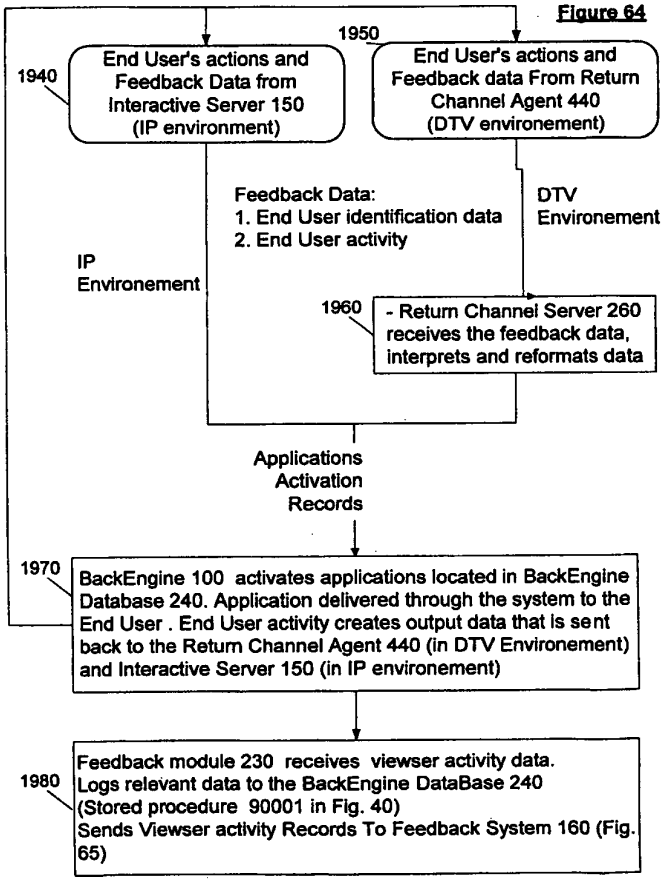


Figure 65

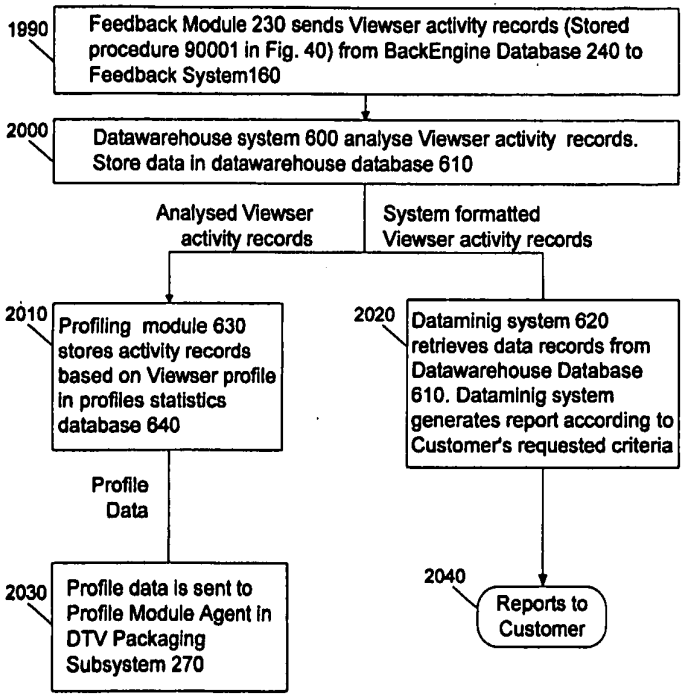


Figure 66A

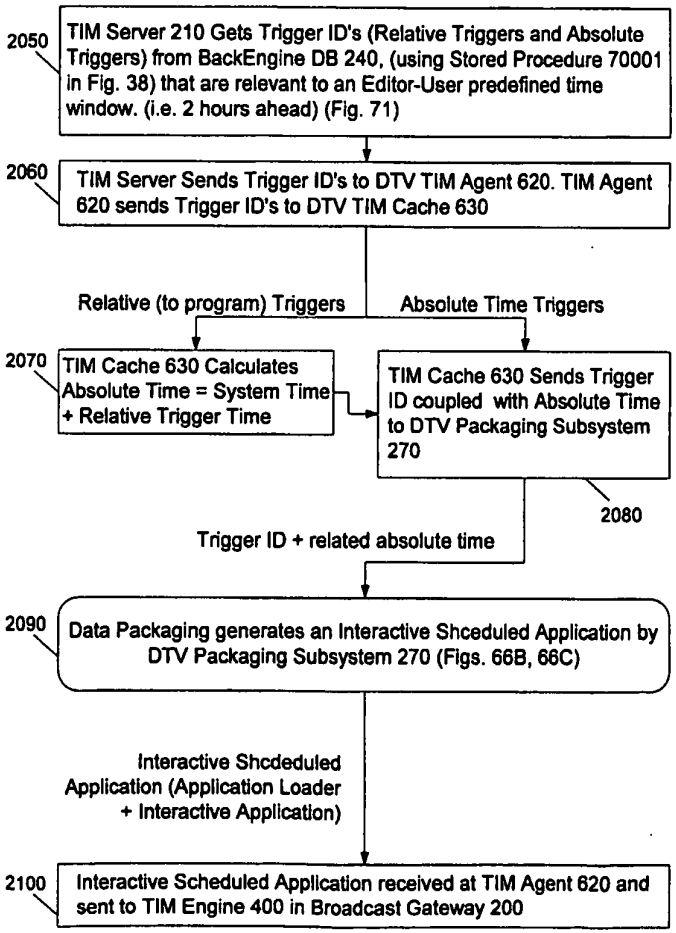
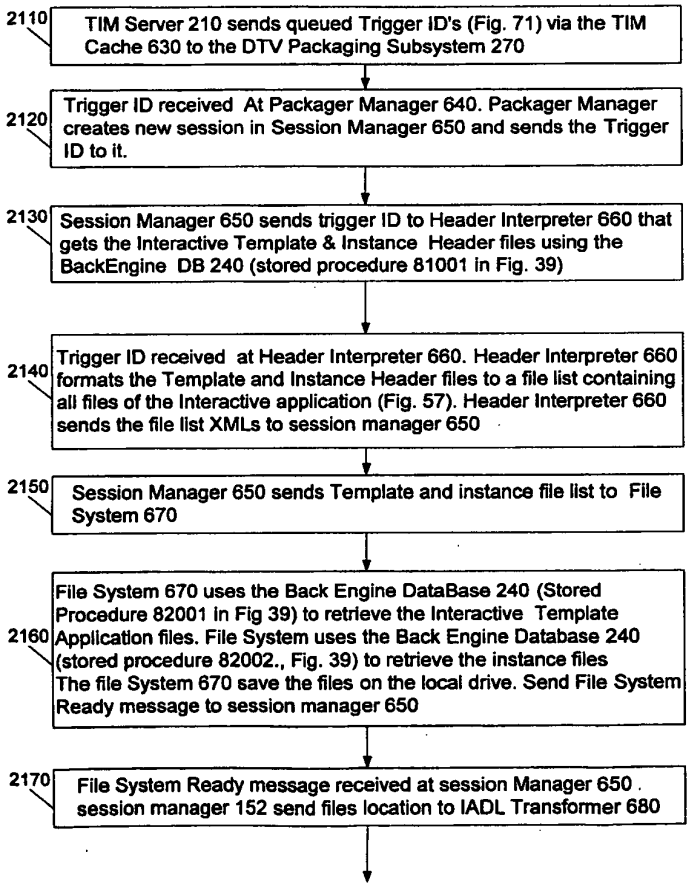


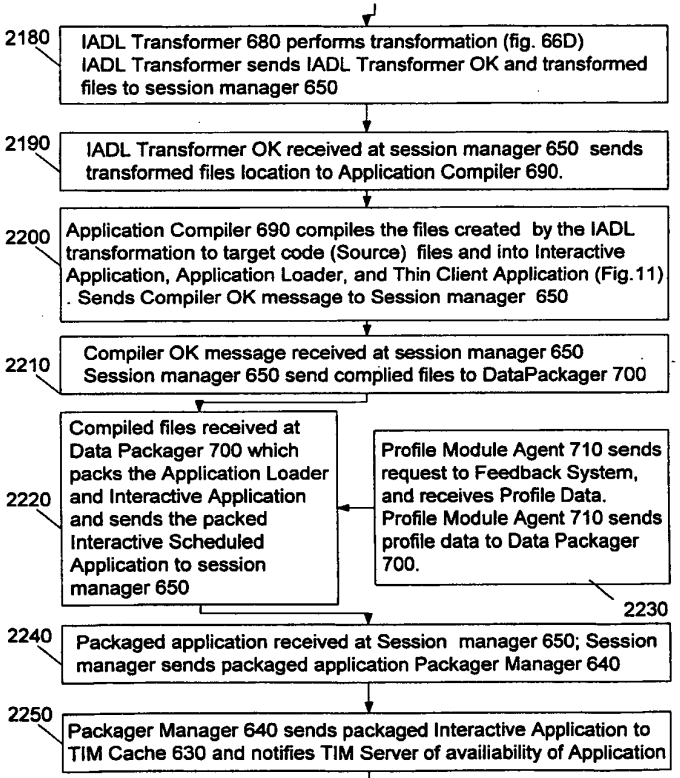
Figure 66B



To Element 2180 in Figure 66C

Figure 66C

From Element 2170 in Figure 66B



To Element 2330 in Figure 67

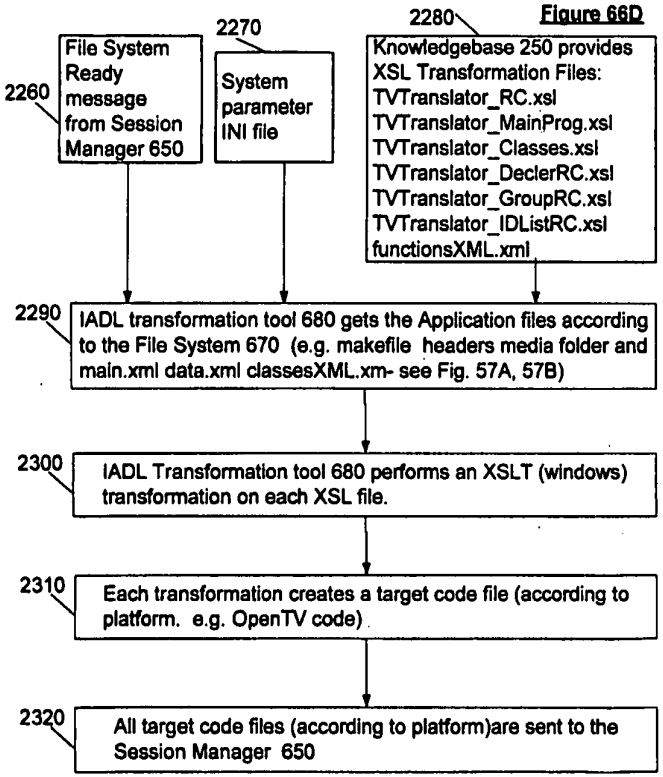


Figure 66D

Figure 67

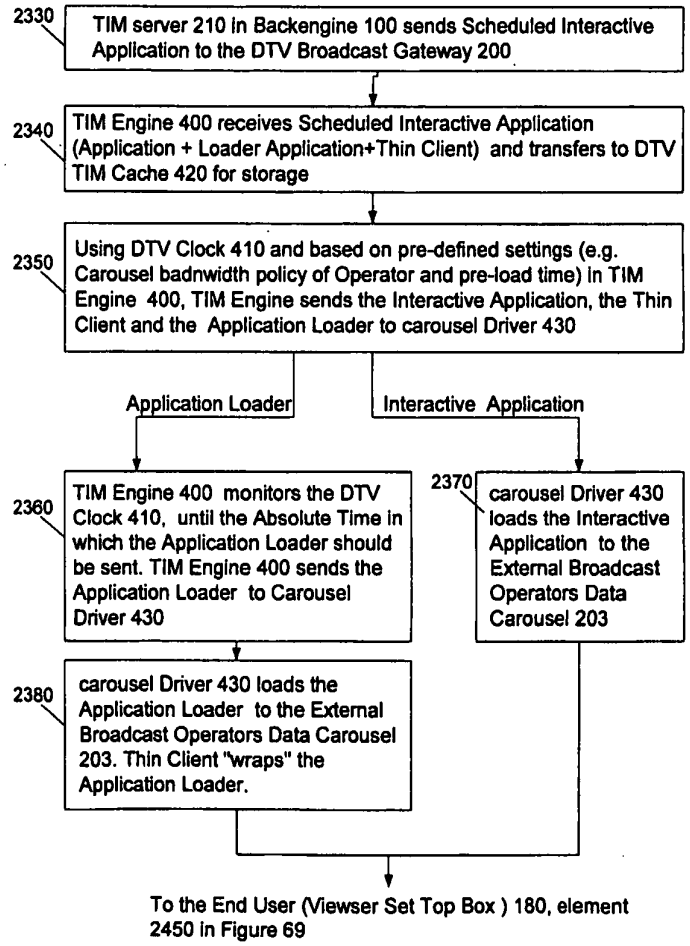


Figure 68

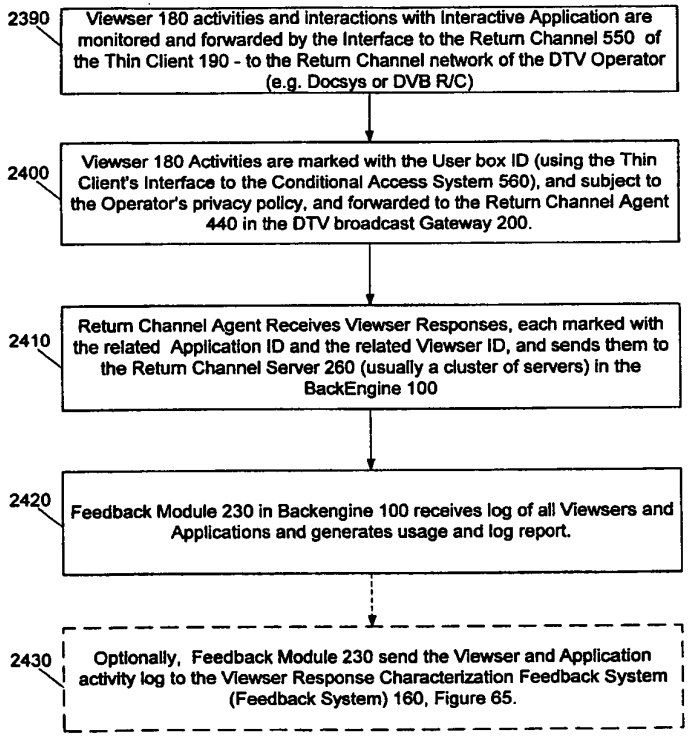


Figure 69

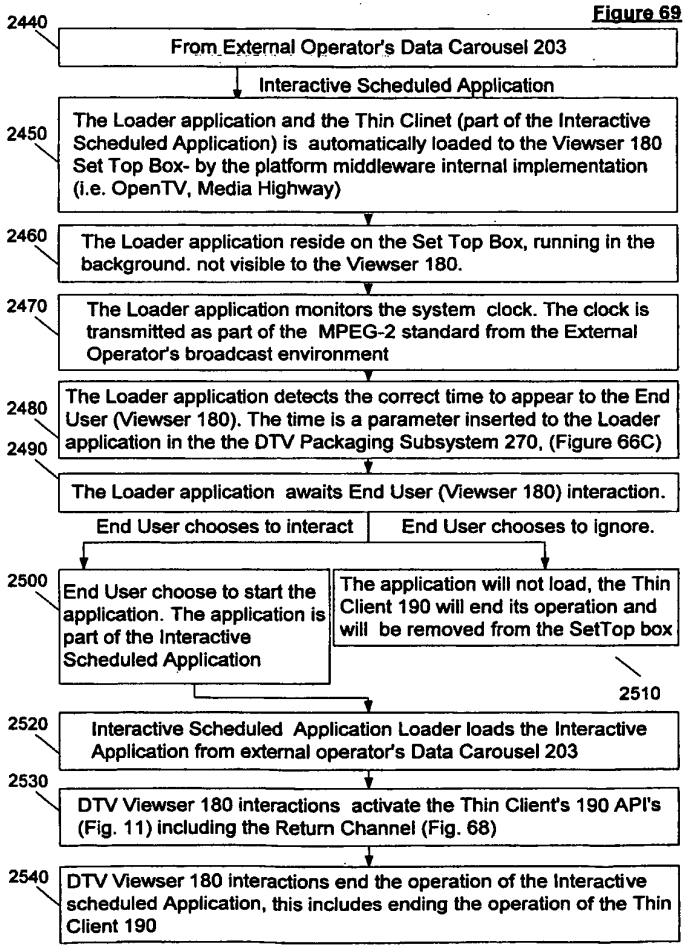


Figure 70

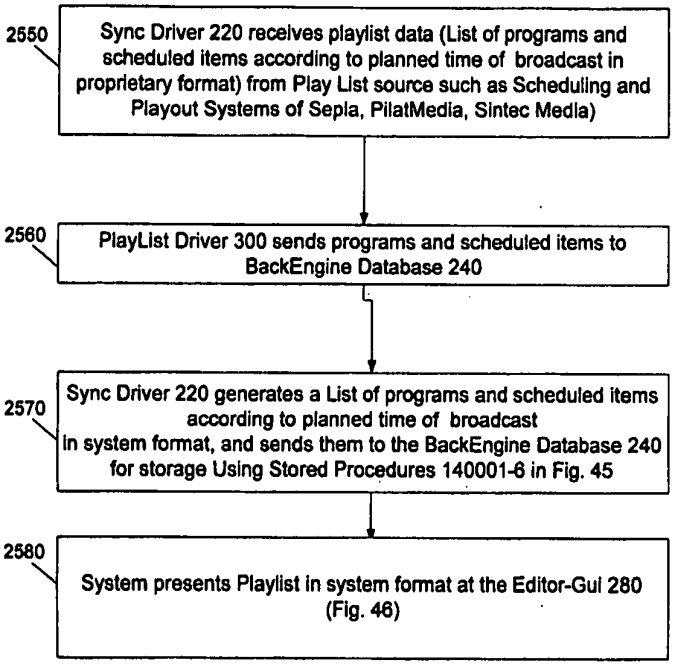


Figure 71A

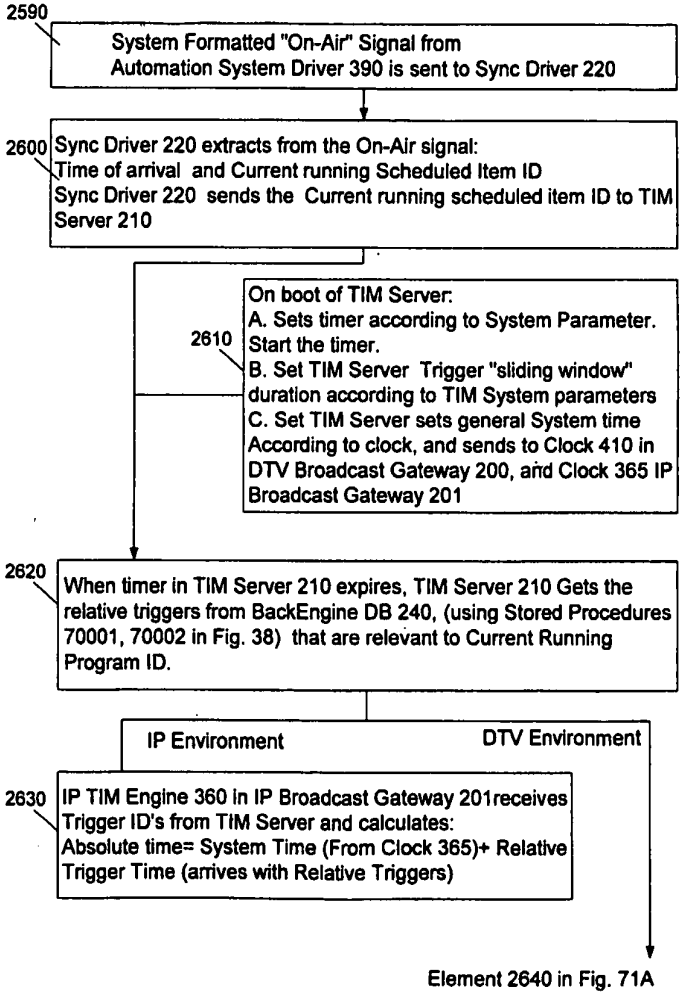


Figure 71B

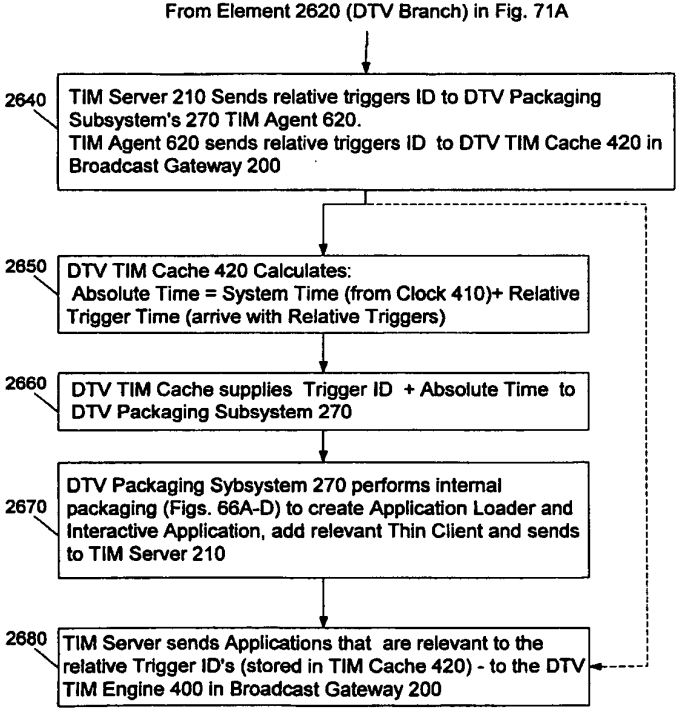


Figure 72A

Applications

Available Attributes for Application Level Syntax:

Attribute name	Value	Description
onLoad	funcName(param)	Describes which functions are executed on onLoad event. Each function can get 0 or more parameters.
onUnload	funcName(param)	Describes which functions are executed on onUnload event. Each function can get 0 or more parameters.

Figure 72B

Applications

Application Level Syntax:

```
<appDoc onLoad="" onUnload="">
<header>
<properties>
<name></name>
<version></version>
<description> </description>
</properties>
<accessories>
</accessories>
</header>
<stage onLoad="" onUnload="" normalClass="" "
uniqueName=""></stage>
<stage onLoad="" onUnload="" normalClass="" "
uniqueName=""></stage>
</appDoc>
```

Figure 73A

Available Attributes for Stage Level Syntax:

Attribute name	Value	Description
normalClass	className	Describes atom's style in normal stage.
onLoad	funcName(param)	Describes which functions are executed on onLoad event. Each function can get 0 or more parameters.
onUnload	funcName(param)	Describes which functions are executed on onUnload event. Each function can get 0 or more parameters.
uniqueName	Id	The unige name of the atom. Connects an atom with function activated from enother object.

Figure 73B

Stages

Stage Level Syntax:

<stage onLoad="" onUnload="" normalClass="

"uniqueName=""></stage>

Figure 74A

Available Attributes for text area element:

Attribute name	Value	Description
onClick	funcName(param)	Describes which functions are executed on onFocus event. Each function can get 0 or more parameters.
onFocus	funcName(param)	Describes which functions are executed on onFocus event. Each function can get 0 or more parameters.
normalClass	className	Describes atom's style in normal stage.
focusClass	className	Describes atom's style in focus stage.
clickClass	className	Describes atom's style in click stage.
isFocus	yes	Describes if the atom is focused when stage loads. Only one object can be focused per stage.
type	keyType	Describes the keyType for activating the onClick event.
uniqueName	Id	The unige name of the atom. Connects an atom with function activated from enother object.
scroll	scrollType	Describes the scroll type in case the data is bigger then the area defined for it. Only one scroll can be defined per stage.

Figure 74B

Text area

Text area Level Syntax:

```
<textArea onClick="" onFocus="" normalClass="" focusClass=""
clickClass="" isFocus="" type="" uniqueName="" scroll="">
      <text onClick="" onFocus="" normalClass="" focusClass=""
      clickClass="" isFocus="" type="" uniqueName="" scroll="">
            text
      </text>
      <text onClick="" onFocus="" normalClass="" focusClass=""
      clickClass="" isFocus="" type="" uniqueName="" scroll="">
            text
      </text>
</textArea>
```


Figure 75A

Available Attributes for line ticker element:

Attribute name	Value	Description
onClick	funcName(param)	Describes which functions are executed on onFocus event. Each function can get 0 or more parameters.
onFocus	funcName(param)	Describes which functions are executed on onFocus event. Each function can get 0 or more parameters.
normalClass	className	Describes atom's style in normal stage.
focusClass	className	Describes atom's style in focus stage.
clickClass	className	Describes atom's style in click stage.
isFocus	yes	Describes if the atom is focused when stage loads. Only one object can be focused per stage.
type	keyType	Describes the keyType for activating the onClick event.
uniqueName	Id	The unige name of the atom. Connects an atom with function activated from enother object.

Figure 75B

Line Ticker

Line Ticker Level Syntax:

```
<lineTicker onClick="" onFocus="" normalClass="" focusClass=""
clickClass="" isFocus="" type="" uniqueName="">
      <text onClick="" onFocus="" normalClass="" focusClass=""
      clickClass="" isFocus="" type="" uniqueName="" scroll="">
            text
      </text>
      <text onClick="" onFocus="" normalClass="" focusClass=""
      clickClass="" isFocus="" type="" uniqueName="" scroll="">
            text
      </text>
</ lineTicker>
```


Figure 76A

Available Attributes for running text area element:

Attribute name	Value	Description
onClick	funcName(param)	Describes which functions are executed on onFocus event. Each function can get 0 or more parameters.
onFocus	funcName(param)	Describes which functions are executed on onFocus event. Each function can get 0 or more parameters.
normalClass	className	Describes atom's style in normal stage.
focusClass	className	Describes atom's style in focus stage.
clickClass	className	Describes atom's style in click stage.
isFocus	yes	Describes if the atom is focused when stage loads. Only one object can be focused per stage.
type	keyType	Describes the keyType for activating the onClick event.
uniqueName	Id	The unige name of the atom. Connects an atom with function activated from enother object.

Figure 76B

Running Text Area

Running Text Area Level Syntax:

```
<runningTextArea  onClick=""  onFocus=""  normalClass=""
focusClass="" clickClass="" isFocus="" type="" uniqueName="">
      <text  onClick=""  onFocus=""  normalClass=""  focusClass=""
      clickClass="" isFocus="" type="" uniqueName="" scroll="">
            text
      </text>
      <text  onClick=""  onFocus=""  normalClass=""  focusClass=""
      clickClass="" isFocus="" type="" uniqueName="" scroll="">
            text
      </text>
</runningTextArea >
```


Figure 77A

Available Attributes for image element:

Attribute name	Value	Description
OnClick	funcName(param)	Describes which functions are executed on onFocus event. Each function can get 0 or more parameters.
OnFocus	funcName(param)	Describes which functions are executed on onFocus event. Each function can get 0 or more parameters.
normalClass	className	Describes atom's style in normal stage.
focusClass	className	Describes atom's style in focus stage.
clickClass	className	Describes atom's style in click stage.
isFocus	yes	Describes if the atom is focused when stage loads. Only one object can be focused per stage.
type	keyType	Describes the keyType for activating the onClick event.
uniqueName	Id	The unige name of the atom. Connects an atom with function activated from enother object.

Figure 77B

Image

Image Level Syntax:

```
<Image onClick="" onFocus="" normalClass="" focusClass=""
clickClass="" isFocus="" type="" uniqueName="">
    <ImageSrc onClick="" onFocus="" normalClass="" focusClass=""
    clickClass="" isFocus="" type="" uniqueName="" focusSrc="">
        file name
    </ImageSrc>
</Image>
```


Figure 78A

Available Attributes for text button elements:

Attribute name	Value	Description
OnClick	funcName(param)	Describes which functions are executed on onFocus event. Each function can get 0 or more parameters.
OnFocus	funcName(param)	Describes which functions are executed on onFocus event. Each function can get 0 or more parameters.
normalClass	className	Describes atom's style in normal stage.
focusClass	className	Describes atom's style in focus stage.
clickClass	className	Describes atom's style in click stage.
isFocus	yes	Describes if the atom is focused when stage loads. Only one object can be focused per stage.
type	keyType	Describes the keyType for activating the onClick event.
uniqueName	Id	The unige name of the atom. Connects an atom with function activated from enother object.

Figure 78B

Text Button

Text Button Level Syntax:

```
<textButton onClick="" onFocus="" normalClass="" focusClass="" clickClass="" isFocus="" type="" uniqueName="">
    <text onClick="" onFocus="" normalClass="" focusClass="" clickClass="" isFocus="" type="" uniqueName="">
        text
    </text>
</ textButton >
```


Figure 79A

Available Attributes for image button elements:

Attribute name	Value	Description
OnClick	funcName(param)	Describes which functions are executed on onFocus event. Each function can get 0 or more parameters.
OnFocus	funcName(param)	Describes which functions are executed on onFocus event. Each function can get 0 or more parameters.
normalClass	ClassName	Describes atom's style in normal stage.
focusClass	ClassName	Describes atom's style in focus stage.
clickClass	ClassName	Describes atom's style in click stage.
IsFocus	Yes	Describes if the atom is focused when stage loads. Only one object can be focused per stage.
Type	KeyType	Describes the keyType for activating the onClick event.
uniqueName	Id	The unige name of the atom. Connects an atom with function activated from enother object.

Figure 79B

Image Button

Image Button Level Syntax:

```
<ImageButton onClick="" onFocus="" normalClass="" focusClass="" clickClass="" isFocus="" type="" uniqueName="">
    <ImageSrc onClick="" onFocus="" normalClass="" focusClass="" clickClass="" isFocus="" type="" uniqueName="" focusSrc="">
        file name
    </ImageSrc>
</ ImageButton >
```


Figure 80A

Available Attributes for Arrows List element:

Attribute name	Value	Description
onClick	funcName(param)	Describes which functions are executed on onFocus event. Each function can get 0 or more parameters.
onFocus	funcName(param)	Describes which functions are executed on onFocus event. Each function can get 0 or more parameters.
normalClass	className	Describes atom's style in normal stage.
focusClass	className	Describes atom's style in focus stage.
clickClass	className	Describes atom's style in click stage.
isFocus	yes	Describes if the atom is focused when stage loads. Only one object can be focused per stage.
type	keyType	Describes the keyType for activating the onClick event.
uniqueName	Id	The unige name of the atom. Connects an atom with function activated from enother object.

Figure 80B

Arrows List

Arrows List Level Syntax:

```
<arrowsList onClick="" onFocus="" normalClass="" focusClass=""
clickClass="" isFocus="" type="" uniqueName="">
      <ImageSrc onClick="" onFocus="" normalClass="" focusClass=""
      clickClass="" isFocus="" type="" uniqueName="" focusSrc="">
            file name
      </ImageSrc>
      <imageSrc onClick="" onFocus="" normalClass="" focusClass=""
      clickClass="" isFocus="" type="" uniqueName="" focusSrc="">
            file name
      </ImageSrc>
      <imageSrc onClick="" onFocus="" normalClass="" focusClass=""
      clickClass="" isFocus="" type="" uniqueName="" focusSrc="">
            file name
      </ImageSrc>
</arrowsList>
```


Figure 81A

Available Attributes for Color List element:

Attribute name	Value	Description
OnClick	funcName(param)	Describes which functions are executed on onFocus event. Each function can get 0 or more parameters.
OnFocus	funcName(param)	Describes which functions are executed on onFocus event. Each function can get 0 or more parameters.
normalClass	className	Describes atom's style in normal stage.
focusClass	className	Describes atom's style in focus stage.
ClickClass	className	Describes atom's style in click stage.
IsFocus	yes	Describes if the atom is focused when stage loads. Only one object can be focused per stage.
Type	keyType	Describes the keyType for activating the onClick event.
uniqueName	Id	The unige name of the atom. Connects an atom with function activated from enother object.

Figure 81B

Color List

Color List Level Syntax:

```
<colorList onClick="" onFocus="" normalClass="" focusClass="" clickClass="" isFocus="" type="" uniqueName="">
    <ImageSrc onClick="" onFocus="" normalClass="" focusClass="" clickClass="" isFocus="" type="" uniqueName="" focusSrc="">
        file name
    </ImageSrc>
    <ImageSrc onClick="" onFocus="" normalClass="" focusClass="" clickClass="" isFocus="" type="" uniqueName="" focusSrc="">
        file name
    </ImageSrc>
    <ImageSrc onClick="" onFocus="" normalClass="" focusClass="" clickClass="" isFocus="" type="" uniqueName="" focusSrc="">
        file name
    </ImageSrc>
</colorList>
```


Figure 82A

Available Attributes for Numbers List element:

Attribute name	Value	Description
OnClick	funcName(param)	Describes which functions are executed on onFocus event. Each function can get 0 or more parameters.
OnFocus	funcName(param)	Describes which functions are executed on onFocus event. Each function can get 0 or more parameters.
NormalClass	className	Describes atom's style in normal stage.
FocusClass	className	Describes atom's style in focus stage.
ClickClass	className	Describes atom's style in click stage.
IsFocus	yes	Describes if the atom is focused when stage loads. Only one object can be focused per stage.
Type	keyType	Describes the keyType for activating the onClick event.
UniqueName	Id	The unige name of the atom. Connects an atom with function activated from enother object.

Figure 82B

Numbers List

Numbers List Level Syntax:

```
<numbersList onClick="" onFocus="" normalClass="" focusClass=""
clickClass="" isFocus="" type="" uniqueName="">
      <imageSrc onClick="" onFocus="" normalClass="" focusClass=""
      clickClass="" isFocus="" type="" uniqueName="" focusSrc="">
            file name
      </imageSrc>
      <imageSrc onClick="" onFocus="" normalClass="" focusClass=""
      clickClass="" isFocus="" type="" uniqueName="" focusSrc="">
            file name
      </imageSrc>
      <imageSrc onClick="" onFocus="" normalClass="" focusClass=""
      clickClass="" isFocus="" type="" uniqueName="" focusSrc="">
            file name
      </imageSrc>
</numbersList>
```


Figure 83A

Available Attributes for Square Shape element:

Attribute name	Value	Description
OnClick	funcName(param)	Describes which functions are executed on onFocus event. Each function can get 0 or more parameters.
OnFocus	funcName(param)	Describes which functions are executed on onFocus event. Each function can get 0 or more parameters.
normalClass	className	Describes atom's style in normal stage.
FocusClass	className	Describes atom's style in focus stage.
ClickClass	className	Describes atom's style in click stage.
IsFocus	yes	Describes if the atom is focused when stage loads. Only one object can be focused per stage.
Type	keyType	Describes the keyType for activating the onClick event.
uniqueName	Id	The unige name of the atom. Connects an atom with function activated from enother object.

Figure 83B

Square Shape

Square Shape Level Syntax:

```
<squareShape onClick="" onFocus="" normalClass="" focusClass=""
clickClass="" isFocus="" type="" uniqueName="">
      <square onClick="" onFocus="" normalClass="" focusClass=""
      clickClass="" isFocus="" type="" uniqueName=""/>
</squareShape>
```


Figure 84A

Available Attributes for Ellipse Shape element:

Attribute name	Value	Description
OnClick	funcName(param)	Describes which functions are executed on onFocus event. Each function can get 0 or more parameters.
OnFocus	funcName(param)	Describes which functions are executed on onFocus event. Each function can get 0 or more parameters.
NormalClass	ClassName	Describes atom's style in normal stage.
FocusClass	ClassName	Describes atom's style in focus stage.
ClickClass	ClassName	Describes atom's style in click stage.
IsFocus	Yes	Describes if the atom is focused when stage loads. Only one object can be focused per stage.
Type	KeyType	Describes the keyType for activating the onClick event.
UniqueName	Id	The unige name of the atom. Connects an atom with function activated from enother object.

Figure 84B

Ellipse Shape

Ellipse Shape Level Syntax:

```
<ellipseShape onClick="" onFocus="" normalClass="" focusClass=""
clickClass="" isFocus="" type="" uniqueName="">
      <square onClick="" onFocus="" normalClass="" focusClass=""
      clickClass="" isFocus="" type="" uniqueName=""/>
</ellipseShape>
```


Figure 85A

Available Attributes for Video Clip element:

Attribute name	Value	Description
OnClick	funcName(param)	Describes which functions are executed on onFocus event. Each function can get 0 or more parameters.
OnFocus	funcName(param)	Describes which functions are executed on onFocus event. Each function can get 0 or more parameters.
normalClass	className	Describes atom's style in normal stage.
FocusClass	className	Describes atom's style in focus stage.
ClickClass	className	Describes atom's style in click stage.
IsFocus	yes	Describes if the atom is focused when stage loads. Only one object can be focused per stage.
Type	keyType	Describes the keyType for activating the onClick event.
uniqueName	Id	The unige name of the atom. Connects an atom with function activated from enother object.

Figure 85B

Video Clip
Video Clip Level Syntax:

```
<videoClip onClick="" onFocus="" normalClass="" focusClass=""
clickClass="" isFocus="" type="" uniqueName="">
      <videoSrc onClick="" onFocus="" normalClass="" focusClass=""
      clickClass="" isFocus="" type="" uniqueName=""/>
</videoClip>
```

Figure 86A

Available Attributes for Timer element:

Attribute name	Value	Description
onClick	funcName(param)	Describes which functions are executed on onFocus event. Each function can get 0 or more parameters.
onFocus	funcName(param)	Describes which functions are executed on onFocus event. Each function can get 0 or more parameters.
normalClass	className	Describes atom's style in normal stage.
focusClass	className	Describes atom's style in focus stage.
clickClass	className	Describes atom's style in click stage.
isFocus	yes	Describes if the atom is focused when stage loads. Only one object can be focused per stage.
Type	keyType	Describes the keyType for activating the onClick event.
uniqueName	Id	The unige name of the atom. Connects an atom with function activated from enother object.

Figure 86B

Timer
Timer Level Syntax:

```
<timer onClick="" onFocus="" normalClass="" focusClass=""
clickClass="" isFocus="" type="" uniqueName="">
    <timer onClick="" onFocus="" normalClass="" focusClass=""
    clickClass="" isFocus="" type="" uniqueName=""/>
</timer>
```


Figure 87A

Available Attributes for text atom

Attribute name	Value	Description
OnClick	funcName(param)	Describes which functions are executed on onFocus event. Each function can get 0 or more parameters.
OnFocus	funcName(param)	Describes which functions are executed on onFocus event. Each function can get 0 or more parameters.
normalClass	className	Describes atom's style in normal stage.
focusClass	className	Describes atom's style in focus stage.
ClickClass	className	Describes atom's style in click stage.
isFocus	yes	Describes if the atom is focused when stage loads. Only one object can be focused per stage.
type	keyType	Describes the keyType for activating the onClick event.
uniqueName	Id	The unige name of the atom. Connects an atom with function activated from enother object.
scroll	scrollType	Describes the scroll type in case the data is bigger then the area defined for it. Only one scroll can be defined per stage.

Figure 87B

Text

Text Level Syntax:

```
<text onClick="" onFocus="" normalClass="" focusClass="" clickClass="" isFocus="" type="" uniqueName="" scroll="">
    text
</text>
```


Figure 88A

Available Attributes for imageSrc atom

Attribute name	Value	Description
OnClick	funcName(param)	Describes which functions are executed on onFocus event. Each function can get 0 or more parameters.
OnFocus	funcName(param)	Describes which functions are executed on onFocus event. Each function can get 0 or more parameters.
normalClass	className	Describes atom's style in normal stage.
focusClass	className	Describes atom's style in focus stage.
ClickClass	className	Describes atom's style in click stage.
IsFocus	yes	Describes if the atom is focused when stage loads. Only one object can be focused per stage.
Type	keyType	Describes the keyType for activating the onClick event.
uniqueName	Id	The unige name of the atom. Connects an atom with function activated from enother object.
FocusSrc	imageName	onFocus image.

Figure 88B

imageSrc
imageSrc Syntax:

```
<ImageSrc onClick="" onFocus="" normalClass="" focusClass="" clickClass="" isFocus="" type="" uniqueName="" focusSrc="">
    file name
</imageSrc>
```

Figure 89A

Available Attributes for videoSrc atom

Attribute name	Value	Description
OnClick	funcName(param)	Describes which functions are executed on onFocus event. Each function can get 0 or more parameters.
OnFocus	funcName(param)	Describes which functions are executed on onFocus event. Each function can get 0 or more parameters.
normalClass	className	Describes atom's style in normal stage.
focusClass	className	Describes atom's style in focus stage.
ClickClass	className	Describes atom's style in click stage.
IsFocus	yes	Describes if the atom is focused when stage loads. Only one object can be focused per stage.
uniqueName	Id	The unige name of the atom. Connects an atom with function activated from enother object.
Type	keyType	Describes the keyType for activating the onClick event.

Figure 89B

videoSrc

videoSrc Syntax:

<videoSrc onClick="" onFocus="" normalClass="" focusClass="" clickClass="" isFocus="" type="" uniqueName="">
file name
</videoSrc>

Figure 90A

Available Attributes for square atom

Attribute name	Value	Description
OnClick	funcName(param)	Describes which functions are executed on onFocus event. Each function can get 0 or more parameters.
OnFocus	funcName(param)	Describes which functions are executed on onFocus event. Each function can get 0 or more parameters.
normalClass	className	Describes atom's style in normal stage.
focusClass	className	Describes atom's style in focus stage.
ClickClass	className	Describes atom's style in click stage.
IsFocus	yes	Describes if the atom is focused when stage loads. Only one object can be focused per stage.
Type	keyType	Describes the keyType for activating the onClick event.
uniqueName	Id	The unige name of the atom. Connects an atom with function activated from enother object.

Figure 90B

Square
Square Syntax:

<square onClick="" onFocus="" normalClass="" focusClass="" clickClass="" isFocus="" type="" uniqueName=""/>

Figure 91A

Available Attributes for ellipse atom

Attribute name	Value	Description
onClick	funcName(param)	Describes which functions are executed on onFocus event. Each function can get 0 or more parameters.
onFocus	funcName(param)	Describes which functions are executed on onFocus event. Each function can get 0 or more parameters.
normalClass	className	Describes atom's style in normal stage.
focusClass	className	Describes atom's style in focus stage.
clickClass	className	Describes atom's style in click stage.
isFocus	yes	Describes if the atom is focused when stage loads. Only one object can be focused per stage.
type	keyType	Describes the keyType for activating the onClick event.
uniqueName	Id	The unige name of the atom. Connects an atom with function activated from enother object.

Figure 91B

Ellipse
Ellipse Syntax:

<ellipse onClick="" onFocus="" normalClass="" focusClass="" clickClass="" isFocus="" type="" uniqueName=""/>

Figure 92A

Available Attributes for soundSrc atom

Attribute name	Value	Description
uniqueName	Id	The unige name of the atom. Connects an atom with function activated from enother object.

Figure 92B

SoundSrc
SoundSrc Syntax:

<soundSrc uniqueName="">
file name
</soundSrc>

Figure 93A

Available Attributes for external link atom

Attribute name	**Value**	**Description**
uniqueName	Id	The unige name of the atom. Connects an atom with function activated from enother object.

Figure 93B

External Link

External Link Syntax:

<externalLink uniqueName="">
URL
</externalLink>

Figure 94A

Available Attributes for char atom

Attribute name	Value	Description
uniqueName	Id	The unige name of the atom. Connects an atom with function activated from enother object.

Figure 94B

Char

Char Syntax:

```
<char uniqueName="">
      character
</char>
```

Figure 95A

Available Attributes for int atom

Attribute name	Value	Description
uniqueName	Id	The unige name of the atom. Connects an atom with function activated from enother object.

Figure 95B

Int
Int Syntax:

```
<Int uniqueName="">
      integer
</Int>
```


Figure 96

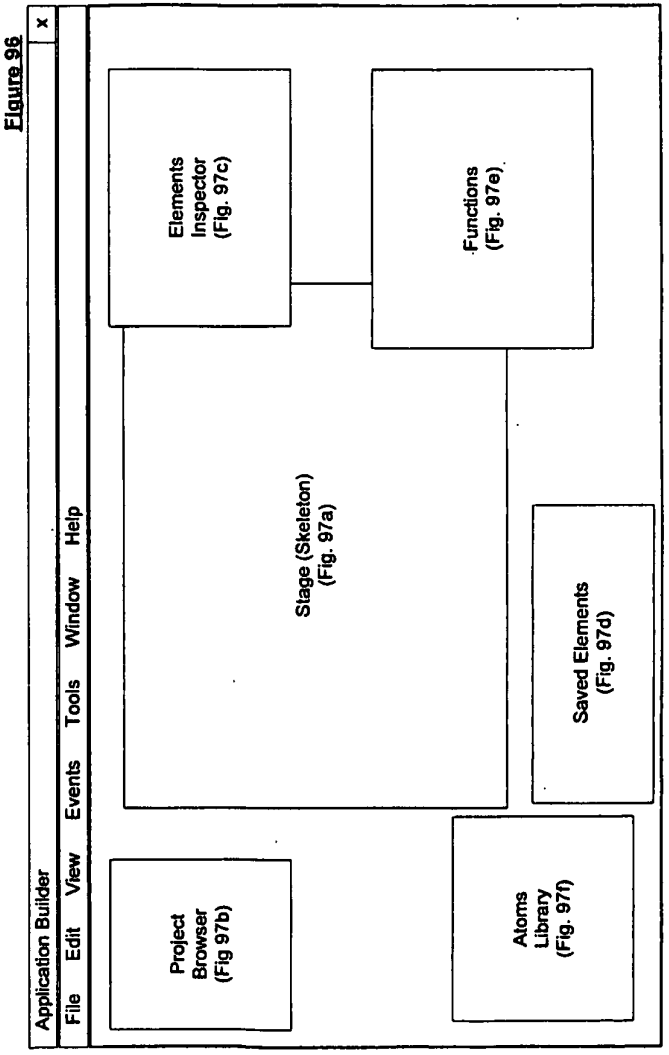


Figure 97a

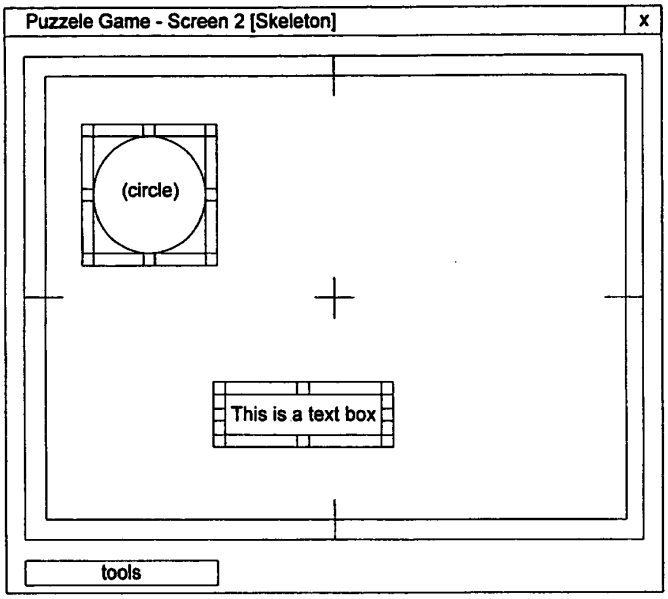


Figure 97b

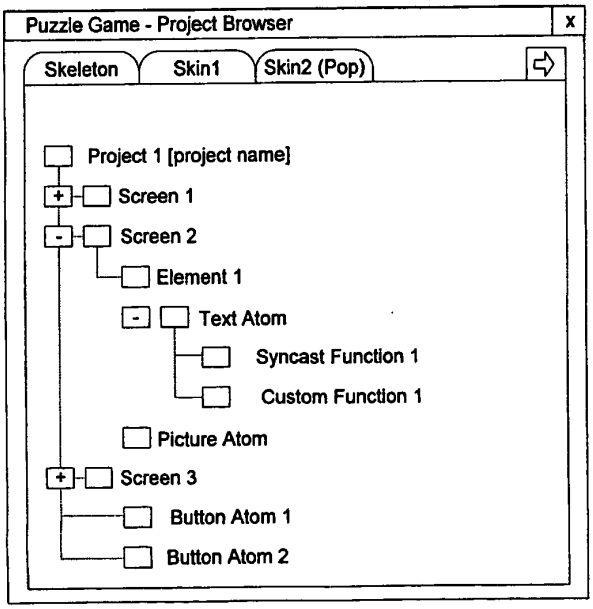


Figure 97c

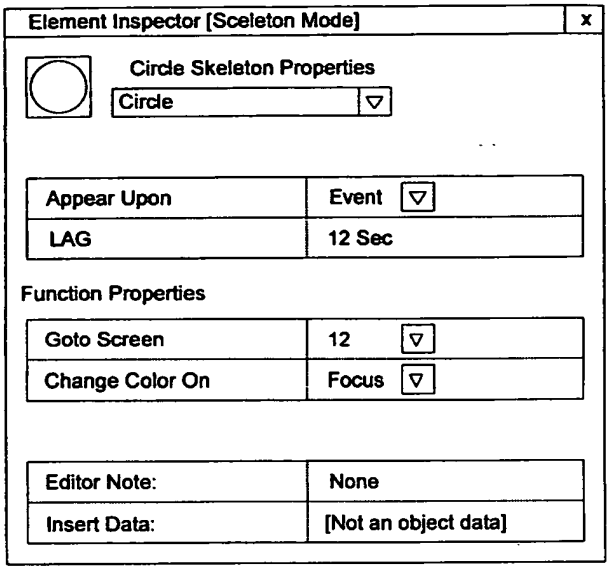


Figure 97d

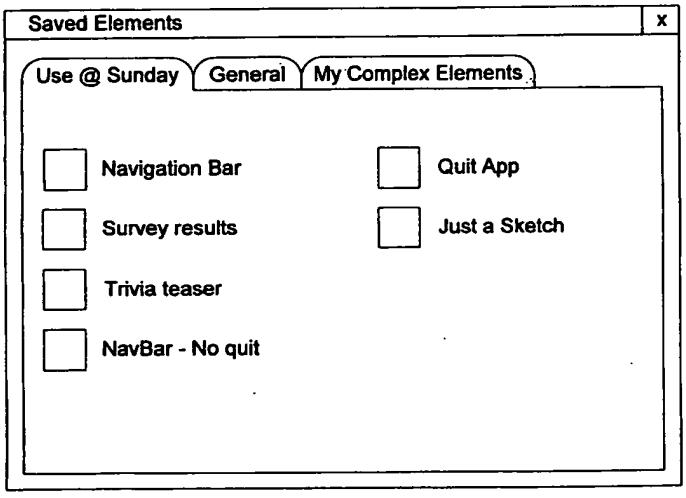


Figure 97e

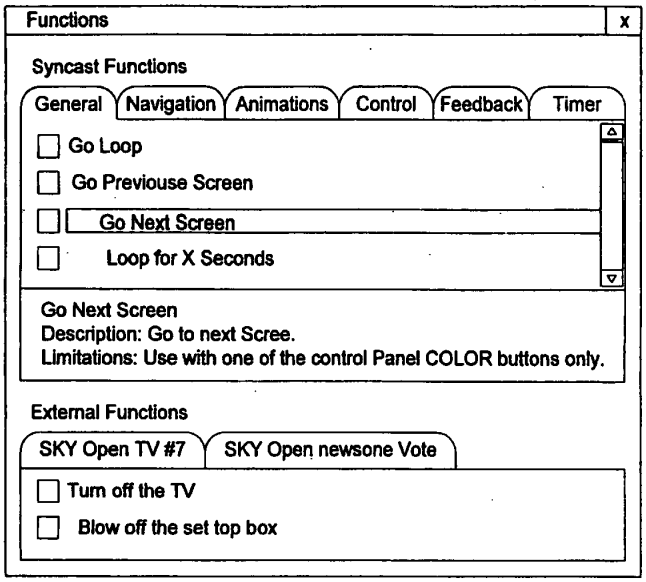
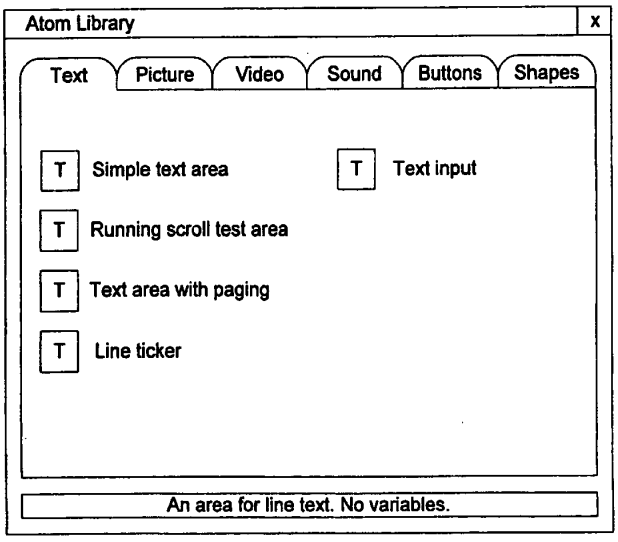


Figure 97f



INTERNATIONAL SEARCH REPORT

International application No.
PCT/IL02/00144

A. CLASSIFICATION OF SUBJECT MATTER

IPC(7) :G06F 3/00; H04N 7/00
US CL : 707/500.1, 501.1, 513; 725/37, 38, 39
According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)

U.S. : 707/500.1, 501.1, 513; 725/37, 38, 39

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)

WEST Database,
search terms: television, TV, programs, guide, electronic television guide, interactive television, EPG, etc.

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	US 5,880,768 A (LEMMONS et al) 09 March 1999, all.	1-23
A, P	US 6,275,648 B1 (KNUDSON et al) 14 August 2001, all.	1-23
A, P	US 6,266,814 B1 (LEMMONS et al) 24 July 2001, all.	1-23
A, P	US 6,323,911 B1 (SCHEIN et al) 27 November 2001, all.	1-23
A, P	US 6,331,877 B1 (BENNINGTON et al) 18 December 2001, all.	1-23
A, E	US 6,388,714 B1 (SCHEIN et al) 14 May 2002, all.	1-23

☐ Further documents are listed in the continuation of Box C. ☐ See patent family annex.

*	Special categories of cited documents:	"T"	later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention
"A"	document defining the general state of the art which is not considered to be of particular relevance	"X"	document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone
"E"	earlier document published on or after the international filing date	"Y"	document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art
"L"	document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)		
"O"	document referring to an oral disclosure, use, exhibition or other means		
"P"	document published prior to the international filing date but later than the priority date claimed	"&"	document member of the same patent family

Date of the actual completion of the international search 18 JUNE 2002	Date of mailing of the international search report 11 JUL 2002
Name and mailing address of the ISA/US Commissioner of Patents and Trademarks Box PCT Washington, D.C. 20231 Facsimile No. (703) 305-3230	Authorized officer STEPHEN HONG Telephone No. (703) 305-9900

Form PCT/ISA/210 (second sheet) (July 1998)*

THIS PAGE BLANK (USPTO)

This Page is Inserted by IFW Indexing and Scanning Operations and is not part of the Official Record

BEST AVAILABLE IMAGES

Defective images within this document are accurate representations of the original documents submitted by the applicant.

Defects in the images include but are not limited to the items checked:

- ☐ BLACK BORDERS
- ☐ IMAGE CUT OFF AT TOP, BOTTOM OR SIDES
- ☑ FADED TEXT OR DRAWING
- ☐ BLURRED OR ILLEGIBLE TEXT OR DRAWING
- ☐ SKEWED/SLANTED IMAGES
- ☐ COLOR OR BLACK AND WHITE PHOTOGRAPHS
- ☐ GRAY SCALE DOCUMENTS
- ☐ LINES OR MARKS ON ORIGINAL DOCUMENT
- ☐ REFERENCE(S) OR EXHIBIT(S) SUBMITTED ARE POOR QUALITY
- ☐ OTHER: _____

IMAGES ARE BEST AVAILABLE COPY.

As rescanning these documents will not correct the image problems checked, please do not report these problems to the IFW Image Problem Mailbox.

THIS PAGE BLANK (USPTO)